A fast algorithm for sparse support vector machines for mobile computing applications

A Fast Algorithm for Sparse Support Vector Machines for Mobile Computing Applications

Jian-Xun Peng, Karen Rafferty, Stuart Ferguson

The School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Ashby Building, Stranmillis Road, Belfast BT9 5AH, UK

Abstract

This research presents a fast algorithm for projected support vector machines (PSVM) by selecting a basis vector set (BVS) for the kernel-induced feature space, the training points are projected onto the subspace spanned by the selected BVS. A standard linear support vector machine (SVM) is then produced in the subspace with the projected training points. As the dimension of the subspace is determined by the size of the selected basis vector set, the size of the produced SVM expansion can be specified. A two-stage algorithm is derived which selects and refines the basis vector set achieving a locally optimal model. The model expansion coefficients and bias are updated recursively for increase and decrease in the basis set and support vector set. The condition for a point to be classed as outside the current basis vector and selected as a new basis vector is derived and embedded in the recursive procedure. This guarantees the linear independence of the produced basis set. The proposed algorithm is tested and compared with an existing sparse primal SVM (SpSVM) and a standard SVM (LibSVM) on seven public benchmark classification problems. Our new algorithm is designed for use in the application area of human activity recognition using smart devices and embedded sensors where their sometimes limited memory and processing resources must be exploited to the full and the more robust and accurate the classification the more satisfied the user. Experimental results demonstrate the effectiveness and efficiency of the proposed algorithm. This work builds upon a previously published algorithm specifically created for activity recognition within mobile applications for the EU Haptimap project [1]. The algorithms detailed in this paper are more memory and resource efficient making them suitable for use with bigger data sets and more easily trained SVMs.

Preprint submitted to Neurocomputing *August 17, 2015*

Keywords: Data classification, sparse support vector machines, recursive algorithm, sequential training algorithm.

1. Introduction

The core aim of the research presented in this paper is to refine and extend a new generation of algorithms to underpin, much more accurately, the process of activity recognition using data derived from mobile sources. As the popularity of low cost portable hand-held computers and mobile phones increases, opportunities for novel context aware applications have grown. Mobile phones can be used along with wearable accelerometers to create valid and reliable measures of physical activity. However, to do this effective algorithms are also needed to interpret the data in the context of different activities. We do this by extending the algorithms published previously [1] which were tested on activity data collected from mobile sensors. Here we improve upon those algorithms and more thoroughly test them (using standard benchmarks) in terms of memory efficiency which is crucial for the mobile storage devices typically used for assisted living.

The algorithms we have developed are based on the Support vector machine(SVM). SVMs are a set of empirical data modelling techniques, which are firmly grounded in the *VC theory* proposed by Vapnik [2], and provide the start-of-the-art performance. The structural risk minimization (SRM) principle implemented by SVM overcomes the difficulties with generalization that have been suffered by traditional neural networks [3], and allows SVMs to provide very accurate solutions.

There has been increasing interest in seeking sparse representations of regular (accurate) SVMs to tackle this problem. Existing techniques proposed for reduced sizes of SVMs fall into two classes: *post-training algorithms* and algorithms that directly yield sparse SVMs,referred to as *sparse algorithms* or *direct algorithms*. Since the generalization performance of a regular SVM is guaranteed (by the SRM principle), post-training algorithms produce a standard SVM in advance and then approximate the normal vector to the separating hyperplane in the feature space, where the SVM discriminant function is expressed as a linear expansion of the support vectors (SVs). A family of linear expansions in the feature space of smaller sizes are used to approximate the normal vector which minimizes the Euclidean distance between the approximation normal and the original one in the feature space is

identified. The approximated SVM discriminant function is thus expressed as the inner product of the approximating normal vector and an input vector in the feature space.

Downs [4] proposed an exact algorithm which prunes SVs from the full SVM solution. Given that the normal vector is a linear combination of the support vectors, all SVs that are linearly dependent (in the feature space) are removed. Obviously, the maximal size of the reduced support vector set is the number of dimensions of the feature space, although it is generally unknown in nonlinear cases. Exact algorithms yield sparse solutions without any loss of the ability to generalize (as the normal vector remains unchanged). However the exact method does not work when a further reduction to the support vector set (SVS) is desired.

Most existing post-training algorithms are approximation methods that approximate the normal vector in a linear expansion of much smaller sizes.

On the other hand sparse SVM algorithms directly minimize the primal objective function with the additional constraint that the normal vector will be a linear expansion of a given number of vectors in the feature space. This means that the search space for the normal vector is restricted, rather than the full feature space required for standard SVMs, and that the resulting reduced size SVM still have a maximal margin.

Lee and Mangasarian [5] randomly choose a subset (typically 1% to 10%) of the given training vectors as candidate SVs during the optimization while classification errors are evaluated over the full training set. In this way the scale of the problem of SVM training (the number of variables) is reduced, resulting in greatly reduced SVM (RSVM) classifiers. However as the expansion vectors are chosen from a random candidate set, and may not be good representatives of the training data, good classification performance can not be guaranteed when the randomly chosen subset is small [6].

Addressing this problem, Wu *et al.* [7][8] proposed algorithms for directly building sparse kernel classifiers. The normal vector to the separating hyperplane is expressed as a linear expansion of a given number of vectors in the feature space. Direct algorithms minimize the primal objective function for standard SVMs with the linear expansion substituted for the normal vector. In addition, the expansion vectors (XVs) for the normal vector are optimized using a gradient-based search, rather than selected from the training set. However optimization of the XVs is a hard non-convex nonlinear problem. Keerthi et al. [9] proposed a sequential incremental algorithm that selects one vector from the training set each time, hence avoiding the hard non-convex

optimization problem for XVs. The corresponding expansion coefficients are optimized using a Newton-Raphson method such that the primal objective function is minimized. This incremental selection is iterated until a given number of vectors are selected.

Typically SVMs are not preferred for real-time applications with limited computational resources (e.g. available RAM or CPU speed)since a large set of support vectors (SVs) is needed to form the SVM classifier, making it computationally complex and expensive to implement. Whilst the advances in sparse SVMs have helped to overcome this issue from a software perspective it is also important to consider the hardware constraints especially when using smart devices. Anguita [10] introduced the concept of a hardware friendly SVM. This method exploits fixed point arithmetic in the feed-forward phase of the SVM. They then extended their models for multiclass problems [11]. They concluded that the use of fixed point calculations is useful in activity recognition applications because they require less memory, processor time and power consumption. Whilst this is important, it is also necessary to refine the basis of the SVM to make it more efficient whether or not it is based on fixed point integers. The main contribution of our paper in terms of algorithmic progress is to project the training points into the subspace spanned by a set of basis vectors selected in the feature space. In the subspace, the SVM is built with the projected training points, referred to as the projected SVM (PSVM). The basis vectors are initially selected from the training set incrementally, and then refined by combining decremental pruning and incremental selecting, resulting in a solution which is optimized over the training set. A condition for a vector that can be selected as an additional basis vector is proposed. This condition is checked recursively in the selection and refining procedures, thus confirming the linear independence of the selected basis set in the feature space. An advantage of PSVM over regular SVM is that the size of the PSVM expansion is determined by the size of the basis set, rather than the number of SVs. Compared with existing sparse algorithms, the SVM does not need to be built in advance and directly minimizes the primal objective function rather than approximating the SVM normal vector. It approaches locally optimal solutions while avoiding hard non-convex non-linear searches. This makes our algorithms particularly suited for implementation on mobile devices and therefore applicable to a range of health-care and assisted living applications.

In Section 2, the PSVM is presented following an outline of regular SVMs in the primal. Section 3 details a sequential algorithm to solve for the PSVM

and optimize the basis set. An implementation of the PSVM algorithm follows in Section 4, where the computational complexity of the algorithm is analyzed. In Section 5, the proposed algorithm is tested over some public benchmark problems and compared with LibSVM for standard SVMs and the sparse SVM algorithm from [9]. Section 6 draws some conclusions on our algorithm. In particular, since in [1] we know the algorithms are suited to activity classification on mobile devices, we use these new benchmark tests to verify the advances made to the algorithms in terms of memory efficiency as clearly they will perform better on the activity classification data they were previously tested on.

2. Projected Support Vector Machines

Given a data set of N point-label pairs $\{(\mathbf{x}_k, y_k), k = 1, \ldots, N\}$, referred to as the training set, each point is represented as a row vector $\mathbf{x}_k \in \Re^{1\times n}$, to which a label of either $+1$ or -1, i.e., $y_k \in \{+1, -1\}$, is attached. This means the training points fall into two categories. This is a binary data classification problem, where a classifier is to be found that can separate the points into two classes. For convenience, the training point set and the associated labels are denoted as $N \times n$ matrix $\mathbf{X} = [\mathbf{x}_1^T, \cdots, \mathbf{x}_N^T]^T$ and $N \times 1$ column vector $\mathbf{y} = [y_1, \cdots, y_N]^T$.

2.1. Regular SVMs in the Primal

Conceptually, a SVM maps its input vector to a high-dimensional space through a kernel-induced map $\phi : \mathbf{x} \to \mathbf{f} = \phi(\mathbf{x})$, where $\mathbf{x} \in \Re^{1\times n}$ is an arbitrary input point to the SVM, $\mathbf{f} = \phi(\mathbf{x})$ denotes its map in the feature space. The high-dimensional space is referred to as the *feature space*, while a point in the feature space, say $\mathbf{f}$, is referred to as a *feature*. In contrast, the space where input vectors $\mathbf{x}$ are from is referred to as the *input space*. Note that $\mathbf{f}$ is represented as a row vector as is $\mathbf{x}$. However the number of dimensions of $\mathbf{f}$ is normally unknown, thus $\mathbf{f}$ cannot be represented numerically.

A SVM defines two parallel hyperplanes $y = \mathbf{f}\mathbf{w} + b \pm 1$ in the feature space that bound the two classes of points in the training set, namely

$$\begin{cases} \mathbf{f}_k\mathbf{w} + b & \geq & +1, \ \texttt{if } y_k = +1 \\ \mathbf{f}_k\mathbf{w} + b & \leq & -1, \ \texttt{if } y_k = -1 \end{cases} \tag{1}$$

hold for $k = 1, \cdots, N$, or equivalently

$$y_k(\mathbf{f}_k\mathbf{w} + b) \geq 1, \ k = 1, \cdots, N \tag{2}$$

The hyperplane $y = \mathbf{f}\mathbf{w} + b$ that lies midway between the two parallel bounding hyperplanes separates the training points, where $\mathbf{f}_k = \phi(\mathbf{x}_k)$ denotes the map of training point $\mathbf{x}_k$ (a point from the input space) in the feature space, $\mathbf{w}$ denotes the normal (column) vector common to the two parallel bounding hyperplanes and the separating hyperplane, and b the interceptor of the separating hyperplane. The interceptors of the two bounding hyperplanes for the two classes $+1$ and -1 are $b + 1$ and $b - 1$, respectively.

For simplicity, denote

$$\left\{ \begin{array}{rcl} e_k & = & 1 - y_k(\mathbf{f}_k\mathbf{w} + b) \\ \xi_k & = & \max(0, e_k) \end{array} \right. , \; k = 1, \cdots, N \tag{3}$$

Hereafter e_k and ξ_k are referred to as the *error* and the *violation* of $\mathbf{x}_k$, respectively. The solution is obtained by maximizing the margin (i.e., $\frac{2}{\sqrt{\mathbf{w}^T\mathbf{w}}}$, the distance between the two parallel bounding hyperplanes) while minimizing the sum of the squared-violations over the training set with regard to $\mathbf{w}$ and b, resulting in the following regularized least-squares (RLS) problem:

$$\min_{\mathbf{w},b} : \quad J(\mathbf{w}, b) = \mathbf{w}^T\mathbf{w} + C\sum_{k=1}^{N} \xi_k^2 \tag{4}$$

where $J(\mathbf{w}, b)$ is the objective function to be minimized, scalar $C > 0$ is referred to as the *penalty parameter*. For a training point, say $\mathbf{x}_k$, that is an outlier of the boundary class, i.e. it violates constraint (2), the violation $\xi_k = e_k = 1 - y_k(\mathbf{f}_k\mathbf{w} + b) > 0$ is penalized in the objective function. For training points that satisfy the constraint (2), it then holds that $e_k \leq 0$ and $\xi_k = 0$, thus no penalty is applied. The penalty parameter C is predefined to balance between the margin and the sum of the squared-violations over the training set. The training points of non-zero violations are referred to as the *support vectors*, and the *support vector set* as a whole.

It is obvious that the objective function $J(\mathbf{w}, b)$ is convex and is continuously differentiable. There is a unique solution of $\mathbf{w}$ and b to problem (4) that causes the first-order partial derivatives $\partial_{\mathbf{w}} J(\mathbf{w}, b)$ and $\partial_b J(\mathbf{w}, b)$ to vanish. The solution is given by

$$\left\{ \begin{array}{rcl} \mathbf{w} & = & \mathbf{F}_S^T\alpha \\ \alpha & = & (C^{-1}\mathbf{I} + \mathbf{K}_S)^{-1}(\mathbf{y}_S - b\mathbf{1}_S) \\ b & = & \frac{\mathbf{1}_S^T(C^{-1}\mathbf{I}+\mathbf{K}_S)^{-1}\mathbf{y}_S}{\mathbf{1}_S^T(C^{-1}\mathbf{I}+\mathbf{K}_S)^{-1}\mathbf{1}_S} \end{array} \right. \tag{5}$$

where subscript S denotes the support vector set, $\mathbf{I}$ denotes the identity matrix of proper size, $\mathbf{K}_S = K(\mathbf{X}_S, \mathbf{X}_S) = \mathbf{F}_S\mathbf{F}_S^T$ denotes the kernel matrix computed by a predefined kernel function $K(\cdot, \cdot)$ over the support vector set represented in matrix $\mathbf{X}_S$, matrix $\mathbf{F}_S = \phi(\mathbf{X}_S)$ represents the maps of the support vectors $\mathbf{X}_S$ in the feature space, $\mathbf{y}_S$ is the column vector collecting the labels of the support vectors, $\mathbf{1}_S$ a column vector of unities of the same size and α is referred to as the SVM *expansion coefficient vector*. Each row of $\mathbf{X}_S$ represents a support vector with its map represented by the corresponding row of $\mathbf{F}_S$. The resulting SVM separating hyperplane is given by

$$f(\mathbf{x}) = \phi(\mathbf{x})\mathbf{w} + b = K(\mathbf{x}, \mathbf{X}_S)\alpha + b \tag{6}$$

where $K(\mathbf{x}, \mathbf{X}_S) = \phi(\mathbf{x})\phi(\mathbf{X}_S)^T$. The label is predicted by the SVM for arbitrary input vector $\mathbf{x}$ as $\hat{y} = sign(f(\mathbf{x}))$.

Generally most of the properties of the kernel-induced map ϕ, even the number of dimensions of the feature space are not known. Therefore the normal vector $\mathbf{w}$ given in (5) is generally unavailable computationally. In this case, the SVM expansion vector α, rather than the normal vector $\mathbf{w}$, is computed.

Looking at (5), as $\mathbf{K}_S$ is symmetric and semi-positive definite and $C > 0$, matrix $C^{-1}\mathbf{I} + \mathbf{K}_S$ is of full rank and thus the matrix inverse $(C^{-1}\mathbf{I} + \mathbf{K}_S)^{-1}$ is well-defined for any training set. It is revealed in (5) that α and b are only dependent on the support vectors. With the solution support vector set (which is unique), α and b are computed using (5). The error given in (3) is rewritten using the kernel as:

$$e_k = 1 - y_k f(\mathbf{x}_k) = 1 - y_k(K(\mathbf{x}_k, \mathbf{X}_S)\alpha + b) \tag{7}$$

It holds that the support vectors have positive errors (non-zero violations) while all other training points have non-positive errors (zero violations). It is easy to verify that in this case the gradient of the objective function vanishes.

It can be noted that the computational complexity of the discriminant function (6) measured by the number of involved kernel function evaluations equals the number of support vectors, and is unknown in advance. It is well known that a SVM can yield a very accurate solution for data classification problems and the generalization performance is theoretically confirmed. However, a large set of support vectors is usually involved in the SVM solution, particularly when the training set is large [12]. This means a computationally complex model which is expensive to implement, is restrictive in

its applications especially for systems with limited computational resources (e.g. available RAM and CPU resources).

In practice it is desirable that, for a given training set, the complexity of the resulting model can be controlled. Addressing this, a set of linearly independent basis vectors is selected from the feature space in this paper. The basis vector set spans a subspace of the full feature space. Each feature (the map of a training point) in the feature space is projected onto this subspace. A linear SVM is then produced in the subspace with the projected features of the training points, instead of features in the full feature space as previously presented. To distinguish the SVM produced in the full feature space, the SVM produced in such a subspace is referred to as a *projected support vector machine* (PSVM). An advantage of PSVMs is that the number of kernel functions involved in the discriminant function is determined by the number of basis vectors rather than the number of support vectors. This makes it possible to balance the complexity and the precision of the model.

2.2. Projected Support Vector Machines

As previously mentioned, points in the feature space, such as the normal vector $\mathbf{w}$ and the features, generally cannot be represented numerically. We cannot select vectors directly from the feature space. Instead, a set of vectors is selected in the input space. These vectors map into the feature space where they are referred to as the basis features and are subsequently used as *basis vectors*.

Suppose a set of B vectors $\bar{\mathbf{x}}_k, k = 1, \cdots, B$ is selected from the input space for the basis. Denote the corresponding basis vectors as $\bar{\mathbf{f}}_k = \phi(\bar{\mathbf{x}}_k), k = 1, \cdots, B$. It is assumed that the basis vectors $\bar{\mathbf{f}}_k$'s are linearly independent. Denote $\mathbf{K}_B = \overline{\mathbf{F}}_B \overline{\mathbf{F}}_B^T = K(\overline{\mathbf{X}}_B, \overline{\mathbf{X}}_B)$. This independence is equivalent to $\mathbf{K}_B$ being fully ranked, namely $rank(\mathbf{X}_B) = B$, where $\overline{\mathbf{X}}_B = [\bar{\mathbf{x}}_1^T, \cdots, \bar{\mathbf{x}}_B^T]^T$ and $\overline{\mathbf{F}}_B = \phi(\overline{\mathbf{X}}_B) = [\bar{\mathbf{f}}_1^T, \cdots, \bar{\mathbf{f}}_B^T]^T$. In the following, we refer to the basis set as $\overline{\mathbf{F}}_B$ or simply $\overline{\mathbf{X}}_B$, and a basis set $\overline{\mathbf{X}}_B$ is said to be linearly independent, if $\phi(\overline{\mathbf{X}}_B)$ is fully ranked in the feature space. Note that in this paper, notations with a bar are for basis vectors; subscripts B and S mean respectively the basis vector set $\overline{\mathbf{X}}_B$ of size B and the support vector set $\mathbf{X}_S$ of size S.

Through the kernel-induced map, each training point, say $\mathbf{x}_k$, produces a feature $\mathbf{f}_k$ (the point map) in the feature space. Projecting $\mathbf{f}_k$ onto the basis vectors produces a coordinate vector, represented as $\mathbf{k}_{B,k} = [\mathbf{f}_k \bar{\mathbf{f}}_1^T, \cdots, \mathbf{f}_k \bar{\mathbf{f}}_B^T] = K(\mathbf{x}_k, \overline{\mathbf{X}}_B)$. In this paper, row vector $\mathbf{k}_{B,k}$ is referred to as the *sub-feature* of training point $\mathbf{x}_k$ (or feature $\mathbf{f}_k$), while the space spanned by the basis vectors

$\overline{\mathbf{F}}_B$ is referred to as the *sub-feature space*, which is a subspace of the feature space.

With the set of sub-features $\mathbf{k}_{B,k}, k = 1, \cdots, N$ corresponding to the training set, a linear SVM can be produced in the sub-feature space as previously done in the full feature space for regular SVMs. Such an SVM produced in the sub-feature space is referred to as a projected SVM (PSVM).

However it should be noted that the selected basis is normally not orthogonal, i.e. $\overline{\mathbf{F}}_B\overline{\mathbf{F}}_B^T = K(\overline{\mathbf{X}}_B, \overline{\mathbf{X}}_B) \neq \mathbf{I}$. The sub-feature space and the feature space have different distance metrics. In other word, different basis sets may have different distance metrics. Looking at the objective function (4), the regularization strength is related to the margin which is the distance between the two bounding hyperplanes. To guarantee a uniform regularization strength during the process of selecting the the basis set, the margin of the PSVM is adjusted as $\mathbf{w}_B^T\overline{\mathbf{F}}_B\overline{\mathbf{F}}_B^T\mathbf{w}_B$ instead of $\mathbf{w}_B^T\mathbf{w}_B$. Note that different sets of basis vectors yield different sets of sub-features, resulting in different PSVMs. The problem of PSVM is to select an optimal set of basis vectors and determine the linear SVM in the sub-feature space (spanned by the basis set) with the set of sub-features corresponding to the given training set, namely

$$\min_{\overline{\mathbf{X}}_B, \mathbf{w}_B, b_B} J(\overline{\mathbf{X}}_B, \mathbf{w}_B, b_B) = \min_{\overline{\mathbf{X}}_B} \min_{\mathbf{w}_B, b_B} J(\overline{\mathbf{X}}_B, \mathbf{w}_B, b_B) \tag{8}$$

The objective function is given by

$$\begin{aligned} J(\overline{\mathbf{X}}_B, \mathbf{w}_B, b_B) &= \mathbf{w}_B^T\mathbf{K}_B\mathbf{w}_B + C\sum_{k=1}^N \xi_k^2 \\ &= \mathbf{w}_B^T\mathbf{K}_B\mathbf{w}_B + C\mathbf{e}^T\mathbf{S}\mathbf{e} \end{aligned} \tag{9}$$

where $\mathbf{w}_B$ and b_B denote the normal vector common to the separating and two bounding hyperplances of the PSVM, $\mathbf{K}_B = \overline{\mathbf{F}}_B\overline{\mathbf{F}}_B^T = K(\overline{\mathbf{X}}_B, \overline{\mathbf{X}}_B)$ is referred to as the *regularization matrix*; e_k and ξ_k are respectively the error and violation of training point $\mathbf{x}_k$, $\mathbf{e} = [e_1, \cdots, e_N]^T$ is the column of errors over the training set, and

$$\begin{aligned} \mathbf{S} &= diag(\mathbf{s}), \mathbf{s} = [s_1, \cdots, s_N]^T \\ s_k &= s(e_k) = \begin{cases} 0, & e_k \leq 0 \\ 1, & e_k > 0 \end{cases} \end{aligned} \tag{10}$$

Matrix $\mathbf{S}$ is an $N \times N$ diagonal matrix with the diagonal entries being N indicators $s_k, k = 1, \cdots, N$ with a '1' denoting that the responding training

point violates the bounding constraint (i.e. of positive error). For support vectors, the errors are positive hence the violations (equal to the corresponding errors) are penalized by unity indicators in $\mathbf{S}$, while for all other training points, the errors are zero or negative hence no violation to be penalized by zero indicators in $\mathbf{S}$. Similar to (3) for regular SVMs, the errors and violations over the training set for the PSVM are given by

$$\left\{ \begin{array}{rcl} e_k & = & 1 - y_k(\mathbf{k}_{B,k}\mathbf{w}_B + b_B) \\ \xi_k & = & \max(0, e_k) \end{array} \right. , \; k = 1, \cdots, N \tag{11}$$

We first investigate the inner optimization of problem (8). For convenience it is written for a given basis set $\overline{\mathbf{X}}_B$ as

$$J(\overline{\mathbf{X}}_B) = \min_{\mathbf{w}_B, b_B} J(\overline{\mathbf{X}}_B, \mathbf{w}_B, b_B) \tag{12}$$

where $J(\overline{\mathbf{X}}_B)$ denotes the minimized value of the objective function (9) for a given basis set $\overline{\mathbf{X}}_B$.

As it is assumed that the basis vector set $\overline{\mathbf{X}}_B$ is linearly independent, matrix $\mathbf{K}_B$ is thus symmetric and positive definite, namely, $\mathbf{K}_B^T = \mathbf{K}_B$ and $\mathbf{K}_B > 0$. This assumption confirms that the objective function $J(\overline{\mathbf{X}}_B, \mathbf{w}_B, b_B)$ is convex and continuously differentiable with regard to $\mathbf{w}_B$ and b_B, and there is a unique solution for $\mathbf{w}_B$ and b_B to the inner problem (12) in which the first-order partial derivatives vanish. The derivates are given by

$$\left\{ \begin{array}{rl} \partial_{\mathbf{w}_B} J & = 2C\mathbf{H}_{B,S}\mathbf{w}_B - 2C\mathbf{K}_{B,S}^T(\mathbf{y}_S - b_B\mathbf{1}_S) \\ \partial_{b_B} J & = 2C\mathbf{1}_S^T(\mathbf{K}_{B,S}\mathbf{w}_B - \mathbf{y}_S + b_B\mathbf{1}_S) \end{array} \right. \tag{13}$$

resulting in the solution of $\mathbf{w}_B$ and b_B to problem (12), given by

$$\left\{ \begin{array}{rcl} \mathbf{w}_B & = & \mathbf{H}_{B,S}^{-1}\mathbf{K}_{B,S}^T(\mathbf{y}_S - b_B\mathbf{1}_S) \\ b_B & = & \frac{\mathbf{1}_S^T(\mathbf{I} - \mathbf{K}_{B,S}\mathbf{H}_{B,S}^{-1}\mathbf{K}_{B,S}^T)\mathbf{y}_S}{\mathbf{1}_S^T(\mathbf{I} - \mathbf{K}_{B,S}\mathbf{H}_{B,S}^{-1}\mathbf{K}_{B,S}^T)\mathbf{1}_S} \end{array} \right. \tag{14}$$

where matrix $\mathbf{K}_{B,S} = \mathbf{F}_S\overline{\mathbf{F}}_B^T = K(\mathbf{X}_S, \overline{\mathbf{X}}_B)$ represents the sub-features of the support vectors indicated by vector $\mathbf{s}$ defined in (10) with each row of $\mathbf{K}_{B,S}$ for a sub-feature. $\mathbf{y}_S$ is the column of the labels of the support vectors $\mathbf{X}_S$, $\mathbf{1}_S$ is a column of unities of the same size as $\mathbf{y}_S$, and

$$\mathbf{H}_{B,S} = C^{-1}\mathbf{K}_B + \mathbf{K}_{B,S}^T\mathbf{K}_{B,S} \tag{15}$$

which is the Hessian (divided by $2C$) of the objective function (8) with regard to the normal vector $\mathbf{w}_B$. Obviously $\mathbf{H}_{B,S}$ is symmetric and positive definite for any linearly independent basis set $\overline{\mathbf{X}}_B$ and any support vector set $\mathbf{X}_S$.

Note that the sub-feature can be computed for an arbitrarily given input vector $\mathbf{x}$ using the kernel function, given by $\phi(\mathbf{x})\phi(\overline{\mathbf{X}}_B)^T = K(\mathbf{x}, \overline{\mathbf{X}}_B) = [K(\mathbf{x}, \bar{\mathbf{x}}_1), \cdots, K(\mathbf{x}, \bar{\mathbf{x}}_B)]$. The sub-features for the training set $\mathbf{k}_{B,k}, k = 1, \cdots, N$ are examples. The discriminant function of the PSVM can be expressed as the inner product of the normal vector $\mathbf{w}_B$ and the sub-feature of $\mathbf{x}$ in the sub-feature space, namely

$$f(\mathbf{x}) = K(\mathbf{x}, \overline{\mathbf{X}}_B)\mathbf{w}_B + b_B \tag{16}$$

Unlike standard SVM where the expansion coefficients α are used instead of the normal vector $\mathbf{w}$ given in (5) which is generally unavailable, PSVM uses the normal vector $\mathbf{w}_B$ directly. In addition, the number of kernel evaluations involved in the discriminant function (6) of standard SVM equals the number of support vectors, while that involved in the PSVM (16) is B (the size of the selected basis vector set $\overline{\mathbf{X}}_B$).

We now investigate the outer minimization of problem (8). For convenience, it is written as follows

$$\min_{\overline{\mathbf{X}}_B} J(\overline{\mathbf{X}}_B) \tag{17}$$

With the optimal $\mathbf{w}_B$ and b_B given in (14), the errors (11) for the support vectors $\mathbf{X}_S$, represented in a column vector, are given by

$$\begin{aligned} \mathbf{e}_S &= \mathbf{1}_S - \mathbf{Y}_S(\mathbf{K}_{B,S}\mathbf{w}_B + b_B\mathbf{1}_S) \\ &= \mathbf{Y}_S[(\mathbf{y}_S - b_B\mathbf{1}_S) - \mathbf{K}_{B,S}\mathbf{w}_B] \\ &= \mathbf{Y}_S(\mathbf{I} - \mathbf{K}_{B,S}\mathbf{H}_{B,S}^{-1}\mathbf{K}_{B,S}^T)(\mathbf{y}_S - b_B\mathbf{1}_S) \end{aligned} \tag{18}$$

where $\mathbf{Y}_S = diag(\mathbf{y}_S)$. As the labels are either $+1$ or -1, we have $\mathbf{Y}_S\mathbf{Y}_S = \mathbf{I}$, $\mathbf{Y}_S\mathbf{1}_S = \mathbf{y}_S$ and $\mathbf{Y}_S\mathbf{y}_S = \mathbf{1}_S$. Noting the optimal bias b_B given in (14), it is easy to verify that

$$\mathbf{y}_S^T\mathbf{e}_S = \mathbf{1}_S^T(\mathbf{I} - \mathbf{K}_{B,S}\mathbf{H}_{B,S}^{-1}\mathbf{K}_{B,S}^T)(\mathbf{y}_S - b_B\mathbf{1}_S) = 0 \tag{19}$$

Given that $J(\overline{\mathbf{X}}_B, \mathbf{w}_B, b_B)$ is convex and continuously differentiable with regard to $\mathbf{w}_B$ and b_B, there are a unique minimizers for $\mathbf{w}_B$ and b_B. $J(\overline{\mathbf{X}}_B)$ defined in (12) is a determined function of the basis set $\overline{\mathbf{X}}_B$. Note again

that diagonal entries of diagonal matrix $\mathbf{S}$, defined in (10), for all the support vectors $\mathbf{X}_S$ are unity, while those for all other training points are zero. Substituting with the minimizer $\mathbf{w}_B$ given in (14) and the corresponding errors of the support vectors $\mathbf{e}_S$ given in (18), results in the minimized objective function $J(\overline{\mathbf{X}}_B)$ defined in (12), and given by $J(\overline{\mathbf{X}}_B) = C\mathbf{1}_S^T\mathbf{e}_S$. With this minimized objective function, the outer optimization problem (17) with regard to $\overline{\mathbf{X}}_B$ is thus rewritten as

$$\min_{\overline{\mathbf{X}}_B} : \quad J(\overline{\mathbf{X}}_B) = C\mathbf{1}_S^T\mathbf{e}_S = \mathbf{y}_S^T\mathbf{Q}_{B,S}^{-1}(\mathbf{y}_S - b_B\mathbf{1}_S) \tag{20}$$

where

$$\mathbf{Q}_{B,S} = C^{-1}\mathbf{I} + \mathbf{K}_{B,S}\mathbf{K}_B^{-1}\mathbf{K}_{B,S}^T \tag{21}$$

As previously mentioned, matrix $\mathbf{K}_B$ is symmetric and is assumed to be positive definite, $\mathbf{Q}_{B,S}$ is also symmetric and positive definite for $C > 0$, hence $\mathbf{Q}_{B,S}^{-1}$ exists. Using the matrix identity

$$(\mathbf{I} + \mathbf{UV})^{-1} = \mathbf{I} - \mathbf{U}(\mathbf{I} + \mathbf{VU})^{-1}\mathbf{V} \tag{22}$$

for matrices $\mathbf{U}$ and $\mathbf{V}$ of proper sizes that confirm the inverses, we have

$$\mathbf{Q}_{B,S}^{-1} = C(\mathbf{I} - \mathbf{K}_{B,S}\mathbf{H}_{B,S}^{-1}\mathbf{K}_{B,S}^T) \tag{23}$$

Looking at (14), it can be noted that the normal vector $\mathbf{w}_B$ and the bias b_B of the PSVM are only dependant on the support vectors. Furthermore, not only the support vectors $\mathbf{X}_S$ but also the associated labels $\mathbf{y}_S$ are involved in the solution (14). In contrast, for the basis vectors $\overline{\mathbf{X}}_B$, there is no requirement for their labels. This is more clearly shown in the following reforming of the objective function $J(\overline{\mathbf{X}}_B)$ by substituting with the minimizer (14) for b_B, given by

$$J(\overline{\mathbf{X}}_B) = \mathbf{y}_S^T(\mathbf{Q}_{B,S}^{-1} - \frac{\mathbf{Q}_{B,S}^{-1}\mathbf{1}_S\mathbf{1}_S^T\mathbf{Q}_{B,S}^{-1}}{\mathbf{1}_S^T\mathbf{Q}_{B,S}^{-1}\mathbf{1}_S})\mathbf{y}_S \tag{24}$$

where $\mathbf{Q}_{B,S}$ given in (21) is defined by the support vector set $\mathbf{X}_S$ along with a given basis vector set $\overline{\mathbf{X}}_B$.

Based on this fact, the basis vectors $\overline{\mathbf{X}}_B$ can be any vectors from the input space, for example, selected from the training set, or more generally, searched from the input space. Selecting $\overline{\mathbf{X}}_B$ from the training set is a subset

selection problem, while searching for the optimal basis vectors $\overline{\mathbf{X}}_B$ in the input space is a continuous nonlinear optimization problem which is a hard non-convex problem. In this paper the basis set is selected from the training set.

Furthermore, solving for the support vector set $\mathbf{X}_S$ and the corresponding $\mathbf{w}_B$ and b_B of the PSVM and optimizing the basis set $\overline{\mathbf{X}}_B$ are two closely coupled problems. Any change in the basis set $\overline{\mathbf{X}}_B$ may result in a different PSVM solution, including the support vector set $\mathbf{X}_S$, $\mathbf{w}_B$ and b_B. This helps ensure that even when only a subspace is selected that resulting SVM also provided a good performance.

In the following section, these two problems are solved alternatively to produce PSVMs with an optimal basis set $\overline{\mathbf{X}}_B$ (of a given size B) selected from the training point set.

3. A Two-stage Algorithm for PSVM

This section presents an algorithm to solve for the PSVM for a given basis set $\overline{\mathbf{X}}_B$, and an algorithm to optimize the basis vector set. The basis vectors are selected from the training set using a subset selection algorithm that combines forward incremental selection (from the training set) and backward decremental pruning. Both optimizations are iterated until a (locally) optimal basis set of given size is approached.

3.1. Solving for the PSVM

The key to solving the inner problem (12) for the PSVM in a sub-feature space (corresponding to a given basis set) is to identify the set of support vectors $\mathbf{X}_S$. As previously discussed, the condition for a set of vectors $\mathbf{X}_S$ being the support vector set that solves the inner minimization problem is given by

$$\begin{cases} e_k > 0 \quad for \quad \forall \mathbf{x}_k \in \mathbf{X}_S \subset \mathbf{X} \\ e_k \leq 0 \quad for \quad \forall \mathbf{x}_k \in \mathbf{X}, \mathbf{x}_k \notin \mathbf{X}_S \end{cases} \tag{25}$$

where $\mathbf{x}_k$ is any one from the training set $\mathbf{X}$, $\mathbf{x}_k \in \mathbf{X}_S$ means that $\mathbf{x}_k$ is a row of $\mathbf{X}_S$ or a support vector, otherwise $\mathbf{x}_k \notin \mathbf{X}_S$. $\mathbf{X}_S \subset \mathbf{X}$ means the support vector set is a subset of the training set.

The basic idea behind the following algorithm is to check the errors of the training points against a given condition (25). For a training point $\mathbf{x}_k$, if $e_k > 0$ and $\mathbf{x}_k \notin \mathbf{X}_S$, $\mathbf{x}_k$ is then added to the support vector set (*increasing the support vector set*). Otherwise if $e_k \leq 0$ and $\mathbf{x}_k \in \mathbf{X}_S$ then it is removed

from the support vector set (*decreasing the support vector set*). For each update of the support vector set, the normal vector $\mathbf{w}_B$, the bias b_B and then the errors $e_k, k = 1, \cdots, N$ are updated accordingly. A new checking iteration is then performed with the updated errors.

Now the current support vector set is denoted as $\mathbf{X}_S$ and the corresponding normal vector and bias as $\mathbf{w}_{B,S}$ and $b_{B,S}$, respectively, where the additional subscript S is to emphasize that $\mathbf{w}_{B,S}$ and $b_{B,S}$ are computed using the support vector set $\mathbf{X}_S$ of size S. Algorithms for increasing and decreasing the support vector set $\mathbf{X}_S$ are presented as follows.

3.1.1. Increasing the support vector set

Suppose a training vector $\mathbf{x}_{S+1} \notin \mathbf{X}_S, e_{S+1} > 0$, i.e., it violates condition (25). It is then added into the support vector set $\mathbf{X}_S$. The increased support vector set is denoted as $\mathbf{X}_{S+1} = [\mathbf{X}_S^T, \mathbf{x}_{S+1}^T]^T$. Assuming that the basis vector set $\overline{\mathbf{X}}_B$ remains unchanged, the sub-features of $\mathbf{X}_{S+1}$ are given by $\mathbf{K}_{B,S+1} = K(\mathbf{X}_{S+1}, \overline{\mathbf{X}}_B) = [\mathbf{K}_{B,S}^T, \mathbf{k}_{B,S+1}^T]^T$ where $\mathbf{k}_{B,S+1} = K(\mathbf{x}_{S+1}, \overline{\mathbf{X}}_B)$ denotes the sub-feature of the new support vector $\mathbf{x}_{S+1}$. From (15), we have $\mathbf{H}_{B,S+1} = C^{-1}\mathbf{K}_B + \mathbf{K}_{B,S+1}^T\mathbf{K}_{B,S+1} = C^{-1}\mathbf{K}_B + \mathbf{K}_{B,S}^T\mathbf{K}_{B,S} + \mathbf{k}_{B,S+1}^T\mathbf{k}_{B,S+1}$ or $\mathbf{H}_{B,S+1} = \mathbf{H}_{B,S} + \mathbf{k}_{B,S+1}^T\mathbf{k}_{B,S+1}$. Using the Woodbury matrix identity [**?**]

$$(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{DA}^{-1}\mathbf{B} + \mathbf{C}^{-1})^{-1}\mathbf{DA}^{-1} \tag{26}$$

matrix inverse $\mathbf{H}_{B,S+1}^{-1}$ is computed recursively from $\mathbf{H}_{B,S}^{-1}$ as follows:

$$\mathbf{H}_{B,S+1}^{-1} = \mathbf{H}_{B,S}^{-1} - \frac{\mathbf{H}_{B,S}^{-1}\mathbf{k}_{B,S+1}^T\mathbf{k}_{B,S+1}\mathbf{H}_{B,S}^{-1}}{\mathbf{k}_{B,S+1}\mathbf{H}_{B,S}^{-1}\mathbf{k}_{B,S+1}^T + 1} \tag{27}$$

Noting that $\mathbf{H}_{B,S}$ is positive definite for any linearly independent basis set $\overline{\mathbf{X}}_B$, (27) is well defined with the denominator $\mathbf{k}_{B,S+1}\mathbf{H}_{B,S}^{-1}\mathbf{k}_{B,S+1}^T + 1 \geq 1$ for any $\mathbf{k}_{B,S+1}$ from the sub-feature space.

To simplify the notation, denote

$$\begin{cases} q_{B,S} &= \mathbf{1}_S^T\mathbf{Q}_{B,S}^{-1}\mathbf{1}_S \\ \mathbf{l}_{B,S} &= \mathbf{H}_{B,S}^{-1}\mathbf{K}_{B,S}^T\mathbf{1}_S \end{cases} \tag{28}$$

Substituting (27) and $\mathbf{K}_{B,S+1} = [\mathbf{K}_{B,S}^T, \mathbf{k}_{B,S+1}^T]^T$ into (14), results in the normal vector $\mathbf{w}_{B,S+1}$ and bias $b_{B,S+1}$ for the increased support vector set $\mathbf{X}_{S+1}$, given by

$$\begin{cases} \mathbf{w}_{B,S+1} &= \mathbf{w}_{B,S} + \eta_{B,S}^+\delta\mathbf{w}_{B,S}^+ \\ b_{B,S+1} &= b_{B,S} + \eta_{B,S}^+(1 - \mathbf{k}_{B,S+1}\mathbf{l}_{B,S}) \end{cases} \tag{29}$$

with

$$\begin{cases} \eta^+_{B,S} = \frac{y_{S+1}e_{S+1/B,S}}{q_{B,S}(\mathbf{k}_{B,S+1}\mathbf{H}^{-1}_{B,S}\mathbf{k}^T_{B,S+1}+1)+(1-\mathbf{k}_{B,S+1}\mathbf{l}_{B,S})^2} \\ \delta\mathbf{w}^+_{B,S} = q_{B,S}\mathbf{H}^{-1}_{B,S}\mathbf{k}^T_{B,S+1} - (1-\mathbf{k}_{B,S+1}\mathbf{l}_{B,S})\mathbf{l}_{B,S} \end{cases} \tag{30}$$

where y_{S+1} is the label of the new support vector $\mathbf{x}_{S+1}$, and $e_{S+1/B,S}$ the error corresponding to the fixed basis set $\overline{\mathbf{X}}_B$ and the original support vectors $\mathbf{X}_S$. As $\mathbf{H}_{B,S}, \mathbf{Q}_{B,S} > 0$ is assumed, we have $q_{B,S} > 0$, and the denominator in (30) satisfies $q_{B,S}(\mathbf{k}_{B,S+1}\mathbf{H}^{-1}_{B,S}\mathbf{k}^T_{B,S+1}+1)+(1-\mathbf{k}_{B,S+1}\mathbf{l}_{B,S})^2 > 0$ for any $\mathbf{x}_{S+1}$.

Using (27) again, it is easy to verify that $q_{B,S}$ and $\mathbf{l}_{B,S}$ defined in (28) can be computed recursively for the increased support vector set $\mathbf{X}_{S+1}$ as follows.

$$\begin{cases} q_{B,S+1} = q_{B,S} + \frac{(1-\mathbf{k}_{B,S+1}\mathbf{l}_{B,S})^2}{\mathbf{k}_{B,S+1}\mathbf{H}^{-1}_{B,S}\mathbf{k}^T_{B,S+1}+1} \\ \mathbf{l}_{B,S+1} = \mathbf{l}_{B,S} + \frac{(1-\mathbf{k}_{B,S+1}\mathbf{l}_{B,S})}{\mathbf{k}_{B,S+1}\mathbf{H}^{-1}_{B,S}\mathbf{k}^T_{B,S+1}+1}\mathbf{H}^{-1}_{B,S}\mathbf{k}^T_{B,S+1} \end{cases} \tag{31}$$

3.1.2. Decreasing the support vector set

Suppose a support vector, say $\mathbf{x}_S \in \mathbf{X}_S, e_S \leq 0$, i.e., it violates condition (25). It is then removed from the support vector set $\mathbf{X}_S$. The decreased support vector set is denoted as $\mathbf{X}_{S-1}$ such that $\mathbf{X}_S = [\mathbf{X}^T_{S-1}, \mathbf{x}^T_S]^T$. Noting $\mathbf{K}_{B,S} = [\mathbf{K}^T_{B,S-1}, \mathbf{k}^T_{B,S}]^T$ where $\mathbf{X}_{S-1} = K(\mathbf{X}_{S-1}, \overline{\mathbf{X}}_B)$ and $\mathbf{k}_{B,S} = K(\mathbf{x}_S, \overline{\mathbf{X}}_B)$, we thus have $\mathbf{H}_{B,S-1} = C^{-1}\mathbf{K}_B + \mathbf{K}^T_{B,S-1}\mathbf{K}_{B,S-1} = \mathbf{H}_{B,S} - \mathbf{k}^T_{B,S}\mathbf{k}_{B,S}$. Using (26) again, the matrix inverse $\mathbf{H}^{-1}_{B,S-1}$ is computed recursively from $\mathbf{H}^{-1}_{B,S}$ as follows.

$$\mathbf{H}^{-1}_{B,S-1} = \mathbf{H}^{-1}_{B,S} - \frac{\mathbf{H}^{-1}_{B,S}\mathbf{k}^T_{B,S}\mathbf{k}_{B,S}\mathbf{H}^{-1}_{B,S}}{\mathbf{k}_{B,S}\mathbf{H}^{-1}_{B,S}\mathbf{k}^T_{B,S} - 1} \tag{32}$$

Given that $\mathbf{H}_{B,S-1} > 0$, vectors $\mathbf{k}_{B,S}$ and $\mathbf{k}^T_{B,S}$ respectively to the left and right of both sides of equation (32) and solving for $\mathbf{k}_{B,S}\mathbf{H}^{-1}_{B,S}\mathbf{k}^T_{B,S}$, results in $\mathbf{k}_{B,S}\mathbf{H}^{-1}_{B,S}\mathbf{k}^T_{B,S} - 1 = -(\mathbf{k}_{B,S}\mathbf{H}^{-1}_{B,S-1}\mathbf{k}^T_{B,S})^{-1} < 0$. This means that (32) is well-defined for any vector $\mathbf{k}_{B,S}$ from the sub-feature space.

Noting $\mathbf{K}_{B,S} = [\mathbf{K}^T_{B,S-1}, \mathbf{k}^T_{B,S}]^T$, substituting (32) into (14) and using notation (28), results in the updated normal vector $\mathbf{w}_{B,S-1}$ and bias $b_{B,S-1}$ for the decreased support vector set $\mathbf{X}_{S-1}$ as follows

$$\begin{cases} \mathbf{w}_{B,S-1} &= \mathbf{w}_{B,S} + \eta^-_{B,S}\delta\mathbf{w}^-_{B,S} \\ b_{B,S-1} &= b_{B,S} + \eta^-_{B,S}(1-\mathbf{k}_{B,S}\mathbf{l}_{B,S}) \end{cases} \tag{33}$$

with

$$\begin{cases} \eta^-_{B,S} = \frac{y_S e_{S/B,S}}{q_{B,S}(\mathbf{k}_{B,S}\mathbf{H}^{-1}_{B,S}\mathbf{k}^T_{B,S}-1)+(1-\mathbf{k}_{B,S}\mathbf{l}_{B,S})^2} \\ \delta\mathbf{w}^-_{B,S} = q_{B,S}\mathbf{H}^{-1}_{B,S}\mathbf{k}^T_{B,S} - (1-\mathbf{k}_{B,S}\mathbf{l}_{B,S})\mathbf{l}^T_{B,S} \end{cases} \tag{34}$$

where y_S is the label of the support vector $\mathbf{x}_S$ being removed, and $e_{S/B,S}$ is the error corresponding to basis set $\overline{\mathbf{X}}_B$ and the original support vector set $\mathbf{X}_S$.

For the decreased support vector set, $q_{B,S-1}$ and $\mathbf{l}_{B,S-1}$ can be computed recursively from $q_{B,S}$ and $\mathbf{l}_{B,S}$ as follows.

$$\begin{cases} q_{B,S-1} &= q_{B,S} + \frac{(1-\mathbf{k}_{B,S}\mathbf{l}_{B,S})^2}{\mathbf{k}_{B,S}\mathbf{H}_{B,S}^{-1}\mathbf{k}_{B,S}^T - 1} \\ \mathbf{l}_{B,S-1} &= \mathbf{l}_{B,S} + \frac{(1-\mathbf{k}_{B,S}\mathbf{l}_{B,S})}{\mathbf{k}_{B,S}\mathbf{H}_{B,S}^{-1}\mathbf{k}_{B,S}^T - 1}\mathbf{H}_{B,S}^{-1}\mathbf{k}_{B,S}^T \end{cases} \tag{35}$$

3.2. Basis Optimization

The basis vector set $\overline{\mathbf{X}}_B$ for the PSVM is optimized using a two step technique. Firstly, an initial set of a given number of basis vectors is selected in a forward incremental way from the training set. Secondly, the initial basis vector set is optimized by combining decremental pruning and incremental adding of basis vectors.

3.2.1. Increasing the Basis Set

Assume one more basis vector $\bar{\mathbf{x}}_{B+1}$ is added to the current basis set $\overline{\mathbf{X}}_B$, forming the increased basis set $\overline{\mathbf{X}}_{B+1} = [\overline{\mathbf{X}}_B^T, \bar{\mathbf{x}}_{B+1}^T]^T$. Correspondingly, the sub-features are increased by one dimension, namely

$$\mathbf{K}_{B+1,S} = [\mathbf{K}_{B,S}, \mathbf{k}_{B+1,S}] \tag{36}$$

where $\mathbf{K}_{B,S} = K(\mathbf{X}_S, \overline{\mathbf{X}}_B)$ and $\mathbf{k}_{B+1,S} = K(\mathbf{X}_S, \bar{\mathbf{x}}_{B+1})$. $\mathbf{K}_{B+1,S}$ is the sub-features of the current support vectors $\mathbf{X}_S$ for the increased basis set $\overline{\mathbf{X}}_{B+1}$, $\mathbf{k}_{B+1,S} = K(\mathbf{X}_S, \bar{\mathbf{x}}_{B+1})$ represents the projects of support vectors $\mathbf{X}_S$ on the new basis vector $\phi(\overline{\mathbf{X}}_{B+1})$. Correspondingly

$$\mathbf{K}_{B+1} = \begin{bmatrix} \mathbf{K}_B & \mathbf{k}_{B+1} \\ \mathbf{k}_{B+1}^T & k_{B+1} \end{bmatrix} \tag{37}$$

and $\mathbf{H}_{B+1,S}$ for the increased basis set $\bar{\mathbf{x}}_{B+1}$ is given by a block matrix as follows

$$\begin{aligned} \mathbf{H}_{B+1,S} &= C^{-1}\mathbf{K}_{B+1} + \mathbf{K}_{B+1,S}^T\mathbf{K}_{B+1,S} \\ &= \begin{bmatrix} \mathbf{H}_{B,S} & \mathbf{h}_{B+1,S} \\ \mathbf{h}_{B+1,S}^T & h_{B+1,S} \end{bmatrix} \end{aligned} \tag{38}$$

where $\mathbf{H}_{B,S} = C^{-1}\mathbf{K}_B + \mathbf{K}_{B,S}^T\mathbf{K}_{B,S}$ is for the basis set $\overline{\mathbf{X}}_B$ before being increased and

$$\begin{cases} \mathbf{h}_{B+1,S} &= C^{-1}\mathbf{k}_{B+1} + \mathbf{K}_{B,S}^T\mathbf{k}_{B+1,S} \\ h_{B+1,S} &= C^{-1}k_{B+1} + \mathbf{k}_{B+1,S}^T\mathbf{k}_{B+1,S} \end{cases} \tag{39}$$

with $\mathbf{k}_{B+1} = K(\overline{\mathbf{X}}_B, \bar{\mathbf{x}}_{B+1})$ and $k_{B+1} = K(\bar{\mathbf{x}}_{B+1}, \bar{\mathbf{x}}_{B+1})$. Matrix inverse $\mathbf{H}_{B+1,S}^{-1}$ for the increased basis set $\overline{\mathbf{X}}_{B+1}$ is given by

$$\mathbf{H}_{B+1,S}^{-1} = \begin{bmatrix} \mathbf{H}_{B,S}^{-1} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \frac{\widetilde{\mathbf{h}}_{B+1,S}\widetilde{\mathbf{h}}_{B+1,S}^T}{\widetilde{h}_{B+1,S}} \tag{40}$$

where

$$\begin{cases} \widetilde{h}_{B+1,S} = h_{B+1,S} - \mathbf{h}_{B+1,S}^T \mathbf{H}_{B,S}^{-1} \mathbf{h}_{B+1,S} \\ \widetilde{\mathbf{h}}_{B+1,S} = \begin{bmatrix} \mathbf{H}_{B,S}^{-1}\mathbf{h}_{B+1,S} \\ -1 \end{bmatrix} \end{cases} \tag{41}$$

Substituting (40) and (36) into (14), the normal vector $\mathbf{w}_{B+1,S}$ (with an increased dimension) and the bias $b_{B+1,S}$ for the increased basis set $\overline{\mathbf{X}}_{B+1}$ and the fixed support vector set $\mathbf{X}_S$ are given by

$$\begin{cases} \mathbf{w}_{B+1,S} = & \begin{bmatrix} \mathbf{w}_{B,S} + \bar{\eta}_{B,S}^+ \bar{\delta}\mathbf{w}_{B,S}^+ \\ \bar{\eta}_{B,S}^+ \end{bmatrix} \\ b_{B+1,S} = & b_{B,S} + \bar{\eta}_{B,S}^+ \bar{\lambda}_{B,S}^+ \end{cases} \tag{42}$$

with

$$\begin{cases} \bar{\eta}_{B,S}^+ = \frac{\mathbf{h}_{B+1,S}^T \mathbf{w}_{B,S} - \mathbf{k}_{B+1,S}^T(\mathbf{y}_S - b_{B,S}\mathbf{1}_S)}{\widetilde{h}_{B+1,S} - \bar{\lambda}_{B,S}^+(\mathbf{k}_{B+1,S}^T\mathbf{1}_S - \mathbf{h}_{B+1,S}^T\mathbf{l}_{B,S})} \\ \bar{\delta}\mathbf{w}_{B,S}^+ = \bar{\lambda}_{B,S}^+ \mathbf{l}_{B,S} - \mathbf{H}_{B,S}^{-1}\mathbf{h}_{B+1,S} \\ \bar{\lambda}_{B,S}^+ = q_{B,S}^{-1}(\mathbf{k}_{B+1,S}^T\mathbf{1}_S - \mathbf{h}_{B+1,S}^T\mathbf{l}_{B,S}) \end{cases} \tag{43}$$

where $q_{B,S}$ and $\mathbf{l}_{B,S}$ are defined in (28). The errors over the fixed support vector set $\mathbf{X}_S$ corresponding to the increased basis set $\overline{\mathbf{X}}_{B+1}$ are computed by $\mathbf{e}_{S/B+1,S} = \mathbf{1}_S - \mathbf{Y}_S(\mathbf{K}_{B+1,S}\mathbf{w}_{B+1,S} + b_{B+1,S}\mathbf{1}_S)$, and the objective function $J(\overline{\mathbf{X}}_{B+1})$ in (20) for the increased basis set $\overline{\mathbf{X}}_{B+1}$ is computed as follows

$$\begin{cases} J_{B+1,S} = C\mathbf{1}_S^T\mathbf{e}_{S/B+1,S} = J_{B,S} - \bar{\delta}J_{B,S}^+ \\ \bar{\delta}J_{B,S}^+ = \frac{[\mathbf{h}_{B+1,S}^T \mathbf{w}_{B,S} - \mathbf{k}_{B+1,S}^T(\mathbf{y}_S - b_{B,S}\mathbf{1}_S)]^2}{\widetilde{h}_{B+1,S} - \bar{\lambda}_{B,S}^+(\mathbf{k}_{B+1,S}^T\mathbf{1}_S - \mathbf{h}_{B+1,S}^T\mathbf{l}_{B,S})} \end{cases} \tag{44}$$

where $J_{B+1,S}$ and $J_{B,S}$ denotes the values of the objective function (24) evaluated with the current support vector set $\mathbf{X}_B$ for basis sets $\overline{\mathbf{X}}_{B+1}$ and $\overline{\mathbf{X}}_B$ respectively. $\bar{\delta}J_{B,S}^+$ denotes the reduction in the objective function due to the increase in the basis set $\overline{\mathbf{X}}_B$ by $\bar{\mathbf{x}}_{B+1,S}$.

For the increased basis vector set $\overline{\mathbf{X}}_{B+1}$, $q_{B+1,S}$ and $\mathbf{l}_{B+1,S}$ can be computed recursively from $q_{B,S}$ and $\mathbf{l}_{B,S}$ as follows.

$$\begin{cases} q_{B+1,S} = q_{B,S} - \mu_{B+1,S}(\mathbf{k}_{B+1,S}^T\mathbf{1}_S - \mathbf{h}_{B+1,S}^T\mathbf{l}_{B,S}) \\ \mathbf{l}_{B+1,S} = \begin{bmatrix} \mathbf{l}_{B,S} \\ 0 \end{bmatrix} + \mu_{B+1,S}\begin{bmatrix} -\mathbf{H}_{B,S}^{-1}\mathbf{h}_{B+1,S} \\ 1 \end{bmatrix} \end{cases} \tag{45}$$

where

$$\mu_{B+1,S} = \frac{(\mathbf{k}_{B+1,S}^T \mathbf{1}_S - \mathbf{h}_{B+1,S}^T \mathbf{l}_{B,S})}{\widetilde{h}_{B+1,S}} \tag{46}$$

In the previous derivation, it was shown that the reduction in the objective function due to an increase in the basis set $\overline{\mathbf{X}}_B$ by $\bar{\mathbf{x}}_{B+1,S}$ is determined by the vector $\bar{\mathbf{x}}_{B+1}$ which is added into the current basis set $\overline{\mathbf{X}}_B$. This reduction is denoted as $\delta J_{B,S}^+ = \delta J_{B,S}^+(\bar{\mathbf{x}}_{B+1})$. It is not difficult to prove that $\delta J_{B,S}^+(\bar{\mathbf{x}}_{B+1}) > 0$ if $\bar{\mathbf{x}}_{B+1}$ is linearly independent of the existing basis vectors $\overline{\mathbf{X}}_B$, namely $rank([\phi(\overline{\mathbf{X}}_B)^T, \phi(\bar{\mathbf{x}}_{B+1,S})^T]) = B + 1$.

Using this basis set increasing algorithm, basis vectors of the PSVM are selected incrementally, one each time. To select an additional basis vector for the current basis set $\overline{\mathbf{X}}_B$, reduction $\delta J_{B,S}^+$ is evaluated for all those training points that don't present in the current basis set $\overline{\mathbf{X}}_B$, and the one that gives the maximal reduction $\delta J_{B,S}^+$ is selected as the new basis vector $\bar{\mathbf{x}}_{B+1}$. In the first stage, this forward incremental selection is iterated until a given number of basis vectors are selected, producing an initial basis set. However, a basis set formed in such an forward incremental way is normally not optimal even locally. The initial basis set is therefore further refined in the second stage.

Note that adding a basis vector $\bar{\mathbf{x}}_{B+1}$ into the existing basis set $\overline{\mathbf{X}}_B$ may cause changes in the minimizer support vector set $\mathbf{X}_B$ that solves the PSVM (12), or in other words, the gradient of the objective function with regard to $\mathbf{w}_{B+1,S}$ and $b_{B+1,S}$ vanishes. However this possible change in the support vector set is not considered in the updated normal vector and bias (42). Therefore one needs to identify the minimizer support vector set $\mathbf{X}_S$ and the corresponding $\mathbf{w}_{B+1,S}$ and $b_{B+1,S}$ after each increase in the basis set.

It is now necessary to investigate the invertibility of the matrix $\mathbf{H}_{B,S}$ defined in (15). Rewritten in features, we have $\mathbf{H}_{B,S} = \overline{\mathbf{F}}_B(C^{-1}\mathbf{I}+\mathbf{F}_S^T\mathbf{F}_S)\overline{\mathbf{F}}_B^T$. Given $C > 0$, matrix $(C^{-1}\mathbf{I}+\mathbf{F}_S^T\mathbf{F}_S)$ has full rank for any $\mathbf{F}_S$ (or any support vector set $\mathbf{X}_S$ and any kernel). Obviously matrix $\mathbf{H}_{B,S}$ has the same rank as the basis set $\overline{\mathbf{F}}_B$. It can be concluded that the basis set $\overline{\mathbf{F}}_B$ being linearly independent is equivalent to the matrix $\mathbf{H}_{B,S}$ being invertible. To confirm the linear independency of the increased basis set $\overline{\mathbf{F}}_{B+1}$, one needs only to make sure that, with the new basis vector $\bar{\mathbf{x}}_{B+1}$ added to $\overline{\mathbf{X}}_B$, $\mathbf{H}_{B+1,S}$ is invertible or $\mathbf{H}_{B+1,S} > 0$, given that $\mathbf{H}_{B+1,S} \geq 0$ for any $\mathbf{X}_S$ and $\overline{\mathbf{X}}_{B+1}$.

For this propose, the following theorem proposes a simple method to check the linear independence of the basis set.

Theorem for basis checking: Consider a series of row-vectors $\{\bar{\mathbf{f}}_i, i =$

$1, 2, \cdots\}$, and an arbitrarily given set of S row-vectors represented in matrix $\mathbf{F}_S, S \geq 1$. Each row of $\mathbf{F}_S$ represents a row-vector. The S row-vectors in $\mathbf{F}_S$ have the same size as $\bar{\mathbf{f}}_i$'s. Define a matrix series $\mathbf{H}_{B,S} = C^{-1}\overline{\mathbf{F}}_B\overline{\mathbf{F}}_B^T + \overline{\mathbf{F}}_B\mathbf{F}_S^T\mathbf{F}_S\overline{\mathbf{F}}_B^T = \overline{\mathbf{F}}_B(C^{-1}\mathbf{I} + \mathbf{F}_S^T\mathbf{F}_S)\overline{\mathbf{F}}_B^T$ for $B = 1, 2, \cdots$, where $C \in \Re, C > 0$, and $\overline{\mathbf{F}}_B = [\bar{\mathbf{f}}_1^T, \cdots, \bar{\mathbf{f}}_B^T]^T$. Assume that $\mathbf{H}_{B,S}$ is invertible. Then matrix $\mathbf{H}_{B+1,S} = \overline{\mathbf{F}}_{B+1}(C^{-1}\mathbf{I} + \mathbf{F}_S^T\mathbf{F}_S)\overline{\mathbf{F}}_{B+1}^T$, which is rewritten as a block-matrix

$$\mathbf{H}_{B+1,S} = \begin{bmatrix} \mathbf{H}_{B,S} & \mathbf{h}_{B+1,S} \\ \mathbf{h}_{B+1,S}^T & h_{B+1,S} \end{bmatrix}, \tag{47}$$

is invertible if and only if

$$h_{B+1,S} > \mathbf{h}_{B+1,S}^T\mathbf{H}_{B,S}^{-1}\mathbf{h}_{B+1,S} \tag{48}$$

where

$$\begin{cases} \mathbf{h}_{B+1,S} = \overline{\mathbf{F}}_B(C^{-1}\mathbf{I} + \mathbf{F}_S^T\mathbf{F}_S)\bar{\mathbf{f}}_{B+1}^T \\ h_{B+1,S} = \bar{\mathbf{f}}_{B+1}(C^{-1}\mathbf{I} + \mathbf{F}_S^T\mathbf{F}_S)\bar{\mathbf{f}}_{B+1}^T \end{cases} \tag{49}$$

It is obvious that matrices $\mathbf{H}_{B,S}, B = 1, 2, ...$ are symmetric and semi-positive definite for any $\overline{\mathbf{F}}_B$ and $\mathbf{F}_S$. Therefore $\mathbf{u}^T\mathbf{H}_{B,S}\mathbf{u} \geq 0, B = 1, 2, \cdots$, holds for any non zero vector $\mathbf{u}$ of proper size. Particularly, letting $\mathbf{u} = [-\mathbf{u}^T\mathbf{H}_{B,S}^{-1}, 1]^T$, we have

$$\begin{array}{rl} & [\mathbf{h}_{B+1,S}^T\mathbf{H}_{B,S}^{-1}, -1]\mathbf{H}_{B+1,S}[\mathbf{h}_{B+1,S}^T\mathbf{H}_{B,S}^{-1}, -1]^T \\ = & h_{B+1,S} - \mathbf{h}_{B+1,S}^T\mathbf{H}_{B,S}^{-1}\mathbf{h}_{B+1,S} \geq 0 \end{array} \tag{50}$$

or

$$h_{B+1,S} \geq \mathbf{h}_{B+1,S}^T\mathbf{H}_{B,S}^{-1}\mathbf{h}_{B+1,S} \tag{51}$$

Now assume $\mathbf{H}_{B+1,S}$ is not invertible. As $\mathbf{H}_{B,S}$ is invertible, the last column of $\mathbf{H}_{B+1,S}$ can be expressed as a linear combination of the other B columns. Namely, there must be an $\mathbf{a}$ that solves the following overdetermined linear system

$$\begin{bmatrix} \mathbf{H}_{B,S} \\ \mathbf{h}_{B+1,S}^T \end{bmatrix}\mathbf{a} = \begin{bmatrix} \mathbf{h}_{B+1,S} \\ h_{B+1,S} \end{bmatrix} \tag{52}$$

where $\mathbf{a}$ is a column vector of B entries. With the first component equality, as $\mathbf{H}_{B,S}$ is invertible, we have $\mathbf{a} = \mathbf{H}_{B,S}^{-1}\mathbf{h}_{B+1,S}$. Substituting into the second component equality, results in $\mathbf{h}_{B+1,S}^T\mathbf{H}_{B,S}^{-1}\mathbf{h}_{B+1,S} = h_{B+1,S}$.

Note that $h_{B+1,S} \geq \mathbf{h}_{B+1,S}^T\mathbf{H}_{B,S}^{-1}\mathbf{h}_{B+1,S}$ always holds. $h_{B+1,S} \neq \mathbf{h}_{B+1,S}^T\mathbf{H}_{B,S}^{-1}\mathbf{h}_{B+1,S}$ is equivalent to $h_{B+1,S} > \mathbf{h}_{B+1,S}^T\mathbf{H}_{B,S}^{-1}\mathbf{h}_{B+1,S}$. We can therefore conclude that if $h_{B+1,S} > \mathbf{h}_{B+1,S}^T\mathbf{H}_{B,S}^{-1}\mathbf{h}_{B+1,S}$ then $\mathbf{H}_{B+1,S}^{-1}$ is invertible.

Conversely, assume that $\mathbf{H}_{B+1,S}$ is invertible. As $\mathbf{H}_{B+1,S}$ is always semi-positive definite, we have $\mathbf{H}_{B+1,S} > 0$, or equivalently, $\mathbf{u}^T\mathbf{H}_{B+1,S}\mathbf{u} > 0$ holds for any non zero vector $\mathbf{u}$ of proper size. Particularly, as $\mathbf{H}_{B,S}^{-1}$ exists, it is possible to let $\mathbf{u} = [\mathbf{h}_{B+1,S}^T\mathbf{H}_{B,S}^{-1}, -1]^T$. We have

$$\begin{aligned} & [\mathbf{h}_{B+1,S}^T\mathbf{H}_{B,S}^{-1}, -1]\mathbf{H}_{B+1,S}[\mathbf{h}_{B+1,S}^T\mathbf{H}_{B,S}^{-1}, -1]^T \\ = \; & h_{B+1,S} - \mathbf{h}_{B+1,S}^T\mathbf{H}_{B,S}^{-1}\mathbf{h}_{B+1,S} > 0 \end{aligned} \tag{53}$$

or (48) holds.

Based on the previous theorem, only those training points satisfying condition (48) can be selected as basis vectors. In addition, condition (48) confirms that $\widetilde{h}_{B+1,S} > 0$ therefore the matrix inversion of $\mathbf{H}_{B+1,S}^{-1}$ given in (40) is well-defined.

Furthermore, it should also be noted that, using the recursive equation (45) for $q_{B+1,S}$ from $q_{B,S}$, the denominator for $\bar{\eta}_{B,S}^+$ in (43) and $\overline{\delta}J_{B,S}^+$ in (44) can be rewritten as $\widetilde{h}_{B+1,S}q_{B,S}^{-1}q_{B+1,S}$. Therefore, if $\mathbf{H}_{B,S}$ and $\mathbf{H}_{B+1,S}$ are invertible, then $q_{B,S}, q_{B+1,S} > 0$ holds, and the denominator $\widetilde{h}_{B+1,S}q_{B,S}^{-1}q_{B+1,S} > 0$.

It should be noted that $\delta J_{B,S}^+$ given in (44) is the reduction of the objective function value due to adding a new basis vector $\bar{\mathbf{x}}_{B+1}$ into the current basis vector set $\overline{\mathbf{X}}_B$, which provides an efficient method of a closed form to score a candidate basis vector for basis selection. In the direct sparse SVM algorithm of Keerthi et al. [9], the basis vectors are also selected incrementally one a time. In order to score a candidate basis vector, a few Newton-Raphson-type iterations are performed on the derivatives (13) of the objective function for the optimal normal vector $\mathbf{w}_{B+1,S}$ and bias $b_{B+1,S}$ (*SpSVM-1*, a $B+2$ – dimensional search), or only for the optimal coefficient of the candidate basis vector while all other existing coefficients (and the bias) are fixed (*SpSVM-2*, a 1- dimensional search). The one from a set of candidate basis vectors that gives the minimal objective function value is then selected into the current basis set.

3.2.2. Decreasing the Basis Set

Without loss of generality, suppose the last basis vector $\bar{\mathbf{x}}_B$ is to be removed from the current basis set $\overline{\mathbf{X}}_B$. For any one of the other basis vectors, permutate relevant matrices (low-permutations to $\overline{\mathbf{X}}_B$, $\mathbf{w}_{B,S}$ and $\mathbf{l}_{B,S}$, low- and column-permutations to $\mathbf{H}_{B,S}$) such that the basis vector to be removed is to the bottom, and the following derivations hold.

Denote the decreased basis set as $\overline{\mathbf{X}}_{B-1}$, such that $\overline{\mathbf{X}}_B = [\overline{\mathbf{X}}_{B-1}^T, \bar{\mathbf{x}}_B^T]^T$. The matrix inverse $\mathbf{H}_{B,S}^{-1}$ is partitioned accordingly

$$\mathbf{H}_{B,S}^{-1} = \begin{bmatrix} \mathbf{V}_{B-1,S} & \mathbf{v}_{B,S} \\ \mathbf{v}_{B,S}^T & v_{B,S} \end{bmatrix} \tag{54}$$

where $\mathbf{V}_{B-1,S}$ is the block corresponding to $\overline{\mathbf{X}}_{B-1}$, $v_{B,S}$ is the entry of $\mathbf{H}_{B,S}^{-1}$ corresponding to basis vector $\bar{\mathbf{x}}_B$ which is to be removed and $\mathbf{v}_{B,S}$ is the corresponding column of $\mathbf{H}_{B,S}^{-1}$ without entry $v_{B,S}$. It is easy to verify that the matrix inverse $\mathbf{H}_{B-1,S}^{-1}$ for the decreased basis set $\overline{\mathbf{X}}_{B-1}$ can be computed recursively from $\mathbf{H}_{B,S}^{-1}$ for the current basis set $\overline{\mathbf{X}}_B$ as follows

$$\mathbf{H}_{B-1,S}^{-1} = \mathbf{V}_{B-1,S} - \frac{\mathbf{v}_{B,S}\mathbf{v}_{B,S}^T}{v_{B,S}} \tag{55}$$

Note that as the basis set $\overline{\mathbf{X}}_B$ is assumed fully ranked, $\overline{\mathbf{X}}_{B-1}$ must be full and $\mathbf{H}_{B-1,S}^{-1}$ therefore exists. Furthermore, $v_{B,S} > 0$ holds if $\mathbf{H}_{B,S}$ is invertible. This is obvious from the recursive equation (40) and (41) as $v_{B,S} = \widetilde{h}_{B,S}^{-1}$ and $\widetilde{h}_{B,S} > 0$ holds for any basis vector $\bar{\mathbf{x}}_B$ that makes $\mathbf{H}_{B,S}$ invertible as previously discussed.

Substituting (54) into the solution (14) and noting (55), it can be derived that

$$\begin{cases} \mathbf{w}_{B-1,S} = & [\mathbf{w}_{B,S}]_{1:B-1} - \bar{\eta}_{B,S}^{-}\bar{\delta}\mathbf{w}_{B,S}^{-} \\ b_{B-1,S} = & b_{B,S} + \bar{\eta}_{B,S}^{-}\bar{\lambda}_{B,S}^{-} \end{cases} \tag{56}$$

with

$$\begin{cases} \bar{\eta}_{B,S}^{-} = (v_{B,S} + \bar{\lambda}_{B,S}^{-}[\mathbf{l}_{B,S}]_B)^{-1}[\mathbf{w}_{B,S}]_B \\ \bar{\delta}\mathbf{w}_{B,S}^{-} = \bar{\lambda}_{B,S}^{-}[\mathbf{l}_{B,S}]_{1:B-1} + \mathbf{v}_{B,S} \\ \bar{\lambda}_{B,S}^{-} = q_{B,S}^{-1}[\mathbf{l}_{B,S}]_B \end{cases} \tag{57}$$

where $[\mathbf{w}_{B,S}]_{1:B-1}$ and $[\mathbf{w}_{B,S}]_B$ are the entries of $\mathbf{w}_{B,S}$ corresponding to $\overline{\mathbf{X}}_{B-1}$ and $\bar{\mathbf{x}}_B$, respectively. $[\mathbf{l}_{B,S}]_{1:B-1}$ and $[\mathbf{l}_{B,S}]_B$ are the corresponding entries of $\mathbf{l}_{B,S}$. The objective function value is updated due to the removal of the basis vector $\bar{\mathbf{x}}_B$ as

$$\begin{cases} J_{B-1,S} = C\mathbf{1}_S^T\mathbf{e}_{S/B-1,S} = J_{B,S} + \delta J_{B,S}^{-}(\bar{\mathbf{x}}_B) \\ \delta J_{B,S}^{-}(\bar{\mathbf{x}}_B) = \frac{[\mathbf{w}_{B,S}]_B^2}{v_{B,S}+\bar{\lambda}_{B,S}^{-}[\mathbf{l}_{B,S}]_B} \end{cases} \tag{58}$$

For the decreased basis vector set $\overline{\mathbf{X}}_{B-1}$, $q_{B-1,S}$ and $\mathbf{l}_{B-1,S}$ can be computed recursively using (55) from $q_{B,S}$ and $\mathbf{l}_{B,S}$ as follows.

$$\begin{cases} q_{B-1,S} = q_{B,S} + v_{B,S}^{-1}[\mathbf{l}_{B,S}]_B^2 \\ \mathbf{l}_{B-1,S} = [\mathbf{l}_{B,S}]_{1:B-1} - v_{B,S}^{-1}[\mathbf{l}_{B,S}]_B \mathbf{v}_{B,S} \end{cases} \tag{59}$$

Using an analogous increment/decrement process that was described in [13], a two stage algorithm is proposed as follows to optimize the basis vector set of the PSVM.

In the first stage, an initial basis set is selected incrementally. Each time the vector that is selected from the training set (while outside the current basis set) is the one that satisfies the independent condition (48), while giving the maximal $\delta J_{B,S}^{+}$.

For simplicity, given a basis vector set $\overline{\mathbf{X}}_B$, we define the set of those training vectors, denoted as $\mathbf{X}_{\widetilde{B}}$, each of which is exclusive of the basis vector set $\overline{\mathbf{X}}_B$, while satisfying the independent condition (48), namely,

$$\begin{aligned} \mathbf{X}_{\widetilde{B}} = \{ \quad & \mathbf{x}_k, k = 1, \cdots, N, \mathbf{x}_k \notin \overline{\mathbf{X}}_B, \\ & rank(\phi([\overline{\mathbf{X}}_B, \mathbf{x}_k])) = B + 1\} \end{aligned} \tag{60}$$

which is hereafter referred to as the *candidate set* for $\overline{\mathbf{X}}_B$, where $rank(\phi([\overline{\mathbf{X}}_B, \mathbf{x}_k])) = B + 1$ means that $\mathbf{x}_k$ is linearly independent of $\overline{\mathbf{X}}_B$ in the feature space.

Furthermore, the reduction of the objective function value due to the addition of a new basis vector to the basis set $\overline{\mathbf{X}}_B$ given in (44) is generalized to any possible vector (i.e. linearly independent of $\overline{\mathbf{X}}_B$), denoted as $\delta J_{B,S}^{+}(\mathbf{x}_k)$ for $\mathbf{x}_k$. $\delta J_{B,S}^{+}(\mathbf{x}_k)$ is referred to as the *significance* of $\mathbf{x}_k$ to the basis set $\overline{\mathbf{X}}_B$, which is the reduction of the objective function value if $\mathbf{x}_k$ was added into $\overline{\mathbf{X}}_B$ as the $B+1$ basis vector $\bar{\mathbf{x}}_{B+i}$. A vector of greater significance is said to be more significant.

With regard to the checking linear independence, Downs et al. [4] identified support vectors that are linearly dependent in the feature space using the row reduced echelon by Noble and Daniel [14]. However, this method would be computationally expensive in this implementation. Rather, in this paper, the linear independence of a vector $\mathbf{x}_k$ outside the current basis set $\overline{\mathbf{X}}_B$ is tested on (48), with which only one additional comparison between two qualities $h_{B+1,S}$ and $\mathbf{h}_{B+1,S}^T \mathbf{H}_{B,S}^{-1} \mathbf{h}_{B+1,S}$ given in (39) is required; the two qualities $h_{B+1,S}$ and $\mathbf{h}_{B+1,S}^T \mathbf{H}_{B,S}^{-1} \mathbf{h}_{B+1,S}$ are necessary to compute the significance of $\mathbf{x}_k$, see equations (40) to (44).

In the first stage, for each iteration of the incremental selection procedure, the candidate set $\mathbf{X}_{\widetilde{B}}$, defined in (60), is formed based on (48) for the current basis set $\overline{\mathbf{X}}_B$, and the most significant one from $\mathbf{X}_{\widetilde{B}}$ is identified and added into $\overline{\mathbf{X}}_B$ as the $B+1$ basis vector $\bar{\mathbf{x}}_{B+i}$. Note that corresponding to each increase in the basis set $\overline{\mathbf{X}}_B$, the support vector set $\mathbf{X}_S$ needs to be updated by solving the PSVM with the increased basis set $\overline{\mathbf{X}}_{B+1}$ for the optimal solver support vector set $\mathbf{X}_B$.

The basis set is produced by iterating the incremental selection procedure in the first stage. However, the basis set obtained in the forward incremental selection, referred to as an *initial basis set*, is normally not optimal even locally. Once an initial basis set has been selected in the first stage, it is further refined by a procedure that combines the previously derived basis set increasing and decreasing algorithms in the second stage.

Each removal of a basis vector, say $\bar{\mathbf{x}}_i \in \overline{\mathbf{X}}_B$, will cause an increase in the objective function value given in (58), rewritten as

$$\delta J^{-}_{B,S}(\bar{\mathbf{x}}_i) = \frac{[\mathbf{w}_{B,S}]_i^2}{[\mathbf{H}_{B,S}]_{i,i} + q_{B,S}^{-1}[\mathbf{l}_{B,S}]_i^2} > 0, \forall \bar{\mathbf{x}}_i \in \overline{\mathbf{X}}_B \tag{61}$$

where $[\mathbf{H}_{B,S}]_{i,i}$ refers to the element (i,i) of matrix $\mathbf{H}_{B,S}$, $[\mathbf{w}_{B,S}]_i$ is the element i of vector $\mathbf{w}_{B,S}$ and $[\mathbf{l}_{B,S}]_i$ is the element i of $\mathbf{l}_{B,S}$, corresponding to the basis vector $\bar{\mathbf{x}}_i$ being removed. Hereafter $\delta J^{-}_{B,S}(\bar{\mathbf{x}}_i)$ is referred to as the significance of basis vector $\bar{\mathbf{x}}_i$ to the basis set $\overline{\mathbf{X}}_{B^{i)}}$, where $\overline{\mathbf{X}}_{B^{i)}}$ denotes the decreased basis set when basis vector $\bar{\mathbf{x}}_i$ is removed from $\overline{\mathbf{X}}_B$. As previously mentioned, $q_{B,S}, [\mathbf{H}_{B,S}]_{i,i} > 0$ always holds if $\mathbf{H}_{B,S}$ is invertible, we therefore have $\delta J^{-}_{B,S}(\bar{\mathbf{x}}_i) > 0, \forall \bar{\mathbf{x}}_i \in \overline{\mathbf{X}}_B$.

It should be noted that, for a basis set $\overline{\mathbf{X}}_B$, the increase in the objective function value $\delta J^{-}_{B,S}(\bar{\mathbf{x}}_i)$ given in (61) due to removing $\bar{\mathbf{x}}_i$ from $\overline{\mathbf{X}}_B$ (resulting in the decreased basis set $\overline{\mathbf{X}}_{B^{i)}}$) refers to the same thing as the reduction of the objective function value $\delta J^{+}_{B^{i)},S}(\bar{\mathbf{x}}_i)$ given in (44) due to adding $\bar{\mathbf{x}}_i$ into $\overline{\mathbf{X}}_{B^{i)}}$. It is not difficult to verify that $\delta J^{+}_{B^{i)},S}(\bar{\mathbf{x}}_i) = \delta J^{-}_{B,S}(\bar{\mathbf{x}}_i)$.

Based on this, an algorithm is proposed in this paper to refine the PSVM basis set which is incrementally selected in the first stage, and generally not optimal even locally. This basis set refinement algorithm combines the decremental removing (from $\overline{\mathbf{X}}_B$) and incremental adding (into $\overline{\mathbf{X}}_{B^{i)}}$) basis vectors, such that the objective function value is further reduced while the size of the basis set remains unchanged.

Considering the current basis set $\overline{\mathbf{X}}_B$, identify the candidate set $\mathbf{X}_{\widetilde{B}}$ defined in (60): For each one from the basis set, say $\bar{\mathbf{x}}_i \in \overline{\mathbf{X}}_B$, evaluate the significance $\delta J^-_{B,S}(\bar{\mathbf{x}}_i)$ of $\bar{\mathbf{x}}_i$ given in (61) and temporarily remove $\bar{\mathbf{x}}_i$ from $\overline{\mathbf{X}}_B$, resulting in an intermediate basis set $\overline{\mathbf{X}}^{i)}_B$. Each of the candidates in $\mathbf{X}_{\widetilde{B}}$ is evaluated using (44) based on $\overline{\mathbf{X}}^{i)}_B$, resulting in a significance $\delta J^+_{B^{i)},S}(\mathbf{x}_k), \forall \mathbf{x}_k \in \mathbf{X}_{\widetilde{B}}$. Find the most significant one, namely, $\mathbf{x}_j = \arg\max\{\delta J^+_{B^{i)},S}(\mathbf{x}_k), \forall \mathbf{x}_k \in \mathbf{X}_{\widetilde{B}}\}$. If $\mathbf{x}_j$ satisfies

$$\delta J^+_{B^{i)},S}(\mathbf{x}_j) > \delta J^-_{B,S}(\bar{\mathbf{x}}_i) \tag{62}$$

then replace $\bar{\mathbf{x}}_i$ in $\overline{\mathbf{X}}_B$ using $\mathbf{x}_j$ (i.e., adding $\mathbf{x}_j$ into $\overline{\mathbf{X}}^{i)}_B$). This results in an updated basis set $[\overline{\mathbf{X}}^T_{B^{i)}}, \mathbf{x}_j^T]^T$ of the same size as $\overline{\mathbf{X}}_B$. The reduction of the objective function corresponding to the replacement of basis vector $\bar{\mathbf{x}}_i$ by $\mathbf{x}_j$ is given by $\delta J^+_{B^{i)},S}(\mathbf{x}_j) - \delta J^-_{B,S}(\bar{\mathbf{x}}_i)$. This decremental-pruning-and-incremental-selection procedure for basis set refinement is iterated until no such basis vector exists in the current basis set; replacement of this basis vector by any training point could result in further reduction in the objective function value. In this sense the refined basis set is locally optimized over the training set.

Note that the comparison (62) should be performed with the same support vector set $\mathbf{X}_S$. In other words, the support vector set $\mathbf{X}_S$ is not changed when $\bar{\mathbf{x}}_i$ is temporarily removed from $\overline{\mathbf{X}}_B$. However if a basis vector is replaced by a training point, the support vector set $\mathbf{X}_S$ needs to be updated, i.e. to solve the PSVM problem (12) for the minimiser support vector set $\mathbf{X}_S$, and update the normal vector $\mathbf{w}_{B,S}$ and bias $b_{B,S}$ with the new basis vector set $[\overline{\mathbf{X}}^T_{B^{i)}}, \mathbf{x}_j^T]^T$. In this way both the PSVM support vector set $\mathbf{X}_S$ (due to the inner minimization) and the basis set $\overline{\mathbf{X}}_B$ (due to the outer minimization) are optimized alternatively until both $\mathbf{X}_S$ and $\overline{\mathbf{X}}_B$ are unchanged. Further details of the implementation of the two-stage algorithm are given in the next section.

4. An Implementation of the Two-stage PSVM

The algorithm presented is an iterative method that optimizes the expansion coefficients (including the normal vector $\mathbf{w}_{B,S}$ and the bias $b_{B,S}$, and thus the support vector set $\mathbf{X}_S$) and the basis vector set alternatively, and needs to be initialized. The iteration can be started from an arbitrarily given

basis set $\overline{\mathbf{X}}_B$ and support vector set $\mathbf{X}_S$. In this study, it is initialized with an initial basis set of one basis vector and an initial support vector set containing all the training points, i.e. $B = 1$ and $S = N$. This initialization is detailed as follows.

4.1. An Initialization

To initialize the algorithm, we first investigate the case of the PSVM with one basis vector, denoted as $\overline{\mathbf{X}}_1 = \bar{\mathbf{x}}_1$. Let the initial support vector set $\mathbf{X}_S = \mathbf{X}$, i.e. $S = N$. The full training set $\mathbf{X} = [\mathbf{x}_1^T, \cdots, \mathbf{x}_N^T]^T$ is taken as the initial support vector set, given that as the basis set increases, the support vector set tends to decrease and approaches that of the SVM built in the full feature space.

¿From the definitions (15) and (21), $\mathbf{H}_{1,S}$ and $\mathbf{Q}_{1,S}$ for basis set $\overline{\mathbf{X}}_1$ of one vector and any support vector set $\mathbf{X}_S$ is given by

$$\begin{cases} \mathbf{H}_{1,S} &= C^{-1}K(\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_1) + \mathbf{k}_{1,S}^T\mathbf{k}_{1,S} \\ \mathbf{Q}_{1,S} &= C^{-1}\mathbf{I} + K^{-1}(\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_1)\mathbf{k}_{1,S}\mathbf{k}_{1,S}^T \end{cases} \tag{63}$$

where $\mathbf{k}_{1,S} = K(\mathbf{X}_S, \bar{\mathbf{x}}_1)$ is the column vector of all the kernel values of the basis vector $\bar{\mathbf{x}}_1$ with all the given support vectors $\mathbf{X}_S$. Substituting (63) into (24), results in the objective function value (for the initial basis set $\overline{\mathbf{X}}_1 = \bar{\mathbf{x}}_1$ and arbitrarily given support vector set $\mathbf{X}_S$), given by

$$J(\overline{\mathbf{X}}_1) = \tilde{\mathbf{y}}_S^T\tilde{\mathbf{y}}_S - \eta_{0,S}^{-1}(\tilde{\mathbf{y}}_S^T\tilde{\mathbf{k}}_{1,S})^2 \tag{64}$$

where $\tilde{\mathbf{y}}_S$ denotes the centred column of labels $\mathbf{y}_S$ associated with the given support vector set $\mathbf{X}_S$, $\tilde{\mathbf{k}}_{1,S}$ the centred column $\mathbf{k}_{1,S}$ and $\eta_{0,S} > 0$ is a scalar, which are given by

$$\begin{cases} \tilde{\mathbf{y}}_S &= \mathbf{y}_S - n_{SV}^{-1}(\mathbf{y}_S^T\mathbf{1}_S)\mathbf{1}_S \\ \tilde{\mathbf{k}}_{1,S} &= \mathbf{k}_{1,S} - n_{SV}^{-1}(\mathbf{k}_{1,S}^T\mathbf{1}_S)\mathbf{1}_S \\ \eta_{0,S} &= C^{-1}K(\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_1) + \tilde{\mathbf{k}}_{1,S}^T\tilde{\mathbf{k}}_{1,S} \end{cases} \tag{65}$$

where n_{SV} denotes the size of the support vector set, i.e. the number of rows in $\mathbf{X}_S$. Looking at (64), the first term $\tilde{\mathbf{y}}_S^T\tilde{\mathbf{y}}_S$ is independent of the basis vector $\bar{\mathbf{x}}_1$. Since a bias b_B is assumed for the PSVM in this paper (16), the columns in $\mathbf{y}_S$ and $\mathbf{k}_{1,S}$ are centered. The second term to the right hand side of (64), denoted as

$$\delta J_{0,S}^+ = \eta_{0,S}^{-1}(\tilde{\mathbf{y}}_S^T\tilde{\mathbf{k}}_{1,S})^2 \tag{66}$$

is used to score each from the training set $\mathbf{X}$, and then the one that gains the most scores is selected as the first basis vector, namely

$$\bar{\mathbf{x}}_1 = \arg\max\{\delta J^+_{0,S}(\mathbf{x}), \forall \mathbf{x} \in \mathbf{X}, K(\mathbf{x},\mathbf{x}) > 0\} \tag{67}$$

where $K(\mathbf{x},\mathbf{x}) > 0$ means $\phi(\mathbf{x}) \neq \mathbf{0}$, confirming that the first selected basis vector is not zero in the feature space. With the first basis vector $\bar{\mathbf{x}}_1$ selected, the corresponding normal vector $\mathbf{w}_{1,S}$ and bias $b_{1,S}$ can be computed by substituting (63) into (14), given by

$$\begin{cases} \mathbf{w}_{1,S} &= \eta^{-1}_{0,S}(\tilde{\mathbf{y}}^T_S \tilde{\mathbf{k}}_{1,S}) \\ b_{1,S} &= n^{-1}_{SV}(\mathbf{y}_S - \mathbf{k}_{1,S}\mathbf{w}_{1,S})^T \mathbf{1}_S \end{cases} \tag{68}$$

while the intermediate qualities $q_{1,S}$ and $\mathbf{l}_{1,S}$ defined in (28) for the recursive iteration are computed for the first basis vector $\bar{\mathbf{x}}_1$ as follows

$$\begin{cases} q_{1,S} &= C n_{SV} \eta_{0,S} \mathbf{H}^{-1}_{1,S} \\ \mathbf{l}_{1,S} &= \mathbf{H}^{-1}_{1,S} \mathbf{k}^T_{1,S} \mathbf{1}_S \end{cases} \tag{69}$$

Note that the qualities $\mathbf{H}_{1,S}$, $\mathbf{w}_{1,S}$ and $\mathbf{l}_{1,S}$ are initially scalars for the initial basis set $\overline{\mathbf{X}}_1$ of one vector. The initialization procedure of the PSVM algorithm is detailed as follows.

Procedure 1 - *Initialization*

Step 1 Let $B = 1$, $\mathbf{X}_S = \mathbf{X}$ and $\mathbf{y}_S = \mathbf{y}$, thus $n_{SV} = N$. Center the label column $\mathbf{y}_S$ using (65) for $\tilde{\mathbf{y}}_S$.

Step 2 *Score*: for each training vector $\mathbf{x}_k, k = 1, \cdots, N$, center column $\mathbf{k}^{k)}_{1,S} = K(\mathbf{X}_S, \mathbf{x}_k)$ using (65) for $\tilde{\mathbf{k}}^{k)}_{1,S}$; compute the score (66) and quality $\eta_{0,S}$ in (65) for $\mathbf{x}_k$, denoted respectively as $\eta_{0,S}(\mathbf{x}_k)$ and $\delta J^+_{0,S}(\mathbf{x}_k)$. Note that if $K(\mathbf{x}_k,\mathbf{x}_k) = 0$, simply let $\delta J^+_{0,S}(\mathbf{x}_k) = 0$.

Step 3 *Select*: find the one that gains the most scores as in (67). Denote the resulting training point as $\bar{\mathbf{x}}_1$, and let $\overline{\mathbf{X}}_B = \bar{\mathbf{x}}_1$. Finally compute $\mathbf{H}_{B,S}$ using (63), $\mathbf{w}_{B,S}$ and $b_{B,S}$ using (68), and $\mathbf{l}_{B,S}$ and $q_{B,S}$ using (69).

In procedure 1, it requires $2N$ floating-point operations (FPOs) to center a vector of length N. For each training point $\mathbf{x}_k$, computing $K(\mathbf{x}_k,\mathbf{x}_k)$ and $K(\mathbf{X}_S,\mathbf{x}_k)$ needs N kernel evaluations; each of the two inner products $\tilde{\mathbf{k}}^T_{1,S}\tilde{\mathbf{k}}_{1,S}$ and $\tilde{\mathbf{y}}^T_S\tilde{\mathbf{k}}_{1,S}$ for $\delta J^+_{0,S}$ and $\eta_{0,S}$ given in (65) and (66) takes $2N$ FPOs. The overall computational complexity of the initial procedure is about $2N(2N+1)$ FPOs, and N^2 additional kernel evaluations. Hereafter an FPO refers to an addition, subtraction, multiplication, division or comparison operation with two floating-point numbers.

4.2. Identification of the Support Vector Set

As previously mentioned, for each change in the basis set, including selection of the first basis vector, the solution (minimizer) support vector set is to be identified. In this study, a recursive algorithm based on (25), (29) and (33) is employed, and an implementing procedure is as detailed in the following.

To simplify the description, denote the training points outside the support vector set $\mathbf{X}_S$ as $\mathbf{X}_{\overline{S}}$. Any one of those points, say $\mathbf{x}_k$, is denoted as $\mathbf{x}_k \in \mathbf{X}_{\overline{S}}$ or $\mathbf{x}_k \notin \mathbf{X}_S$, and for a support vector set $\mathbf{X}_S$, define the set of indices of the training points that violate condition (25) as follows

$$\begin{aligned} I(\mathbf{X}_S) = \quad & \{k, k = 1, \cdots, N, e_k \leq 0, \mathbf{x}_k \in \mathbf{X}_S\} \bigcup \\ & \{k, k = 1, \cdots, N, e_k > 0, \mathbf{x}_k \notin \mathbf{X}_S\} \end{aligned} \tag{70}$$

If $I(\mathbf{X}_S) = \emptyset$ (is empty), then the support vector set $\mathbf{X}_S$ solves the PSVM (12) for a given $\overline{\mathbf{X}}_B$, and the minimizer $\mathbf{w}_{B,S}$ and $b_{B,S}$ are computed using (14). In this paper, the solver $\mathbf{X}_S$, along with the minimizers $\mathbf{w}_{B,S}$ and $b_{B,S}$, is identified recursively in the following six steps iterative procedure.

Procedure 2 *- Support Vector Set Identification*

Step 1 Evaluate the errors over the training set using (7) for the current $\mathbf{w}_{B,S}$ and $b_{B,S}$.

Step 2 *Check*: identify the points that violate condition (25), and form the index set I defined in (70), and sort the indices in I by the magnitudes of the errors in descending order. If $I = \emptyset$, then exit; otherwise goto step 3.

Step 3 *Shunt*: for any one element from I, say $k \in I$, if $\mathbf{x}_k \in \mathbf{X}_S$, then go to step 4; otherwise goto step 5.

Step 4 *Decrease*: remove $\mathbf{x}_k$ from $\mathbf{X}_S$, and for the removing of $\mathbf{x}_k$ (instead of $\mathbf{x}_S$) compute $\mathbf{w}_{B,S-1}$ and $b_{B,S-1}$ using (33), $q_{B,S-1}$ and $\mathbf{l}_{B,S-1}$ using (35) and $\mathbf{H}^{-1}_{B,S-1}$ using (32). Finally let $S \leftarrow S - 1$ and goto step 6.

Step 5 *Increase*: add $\mathbf{x}_k$ into $\mathbf{X}_S$, and correspondingly compute $\mathbf{w}_{B,S+1}$ and $b_{B,S+1}$ using (29), $q_{B,S+1}$ and $\mathbf{l}_{B,S+1}$ using (31) and $\mathbf{H}^{-1}_{B,S+1}$ using (27); let $S \leftarrow S + 1$ and finally goto step 6.

Step 6 *Update*: remove k from I. If $I \neq \emptyset$ then update the errors using (7) for the updated $\mathbf{w}_{B,S}$ and $b_{B,S}$ for all the points of which the indices remain in the decreased set I. With the updated errors indexed by I, check against condition (25) and remove all those indices that satisfy (25). If $I = \emptyset$ then goto step 1; otherwise goto step 3.

Using procedure 2, the support vector set that solves PSVM (12) for a given basis set $\overline{\mathbf{X}}_B$ can be identified in very few ($1 \sim 4$ in most cases, even in the initial stage) iterations from step 1 to step 6. In addition, an increase or a decrease in the basis set, or a replacement of a basis vector by a new basis vector, often causes a change in the solution basis set by only a small proportion of support vectors, i.e. the index set I is often very small compared with the support vector set $\mathbf{X}_S$, particularly in the later stage when the support vector set approaches that of the full SVM built in the full feature space.

To evaluate the errors (over the full training set in step 1) requires about $2NB$ FPOs and NB kernel evaluations. To remove a support vector from $\mathbf{X}_S$ in step 4, $\mathbf{w}_{B,S-1}$, $b_{B,S-1}$, $q_{B,S-1}$, $\mathbf{l}_{B,S-1}$ and $\mathbf{H}^{-1}_{B,S-1}$ must be computed using (33), (35) and (32), requiring about $B(3B + 12)$ FPOs, given that $\mathbf{H}^{-1}_{B,S-1}$ is symmetric. To add a support vector to $\mathbf{X}_S$ in step 5, it needs to compute $\mathbf{w}_{B,S+1}$, $b_{B,S+1}$, $q_{B,S+1}$, $\mathbf{l}_{B,S+1}$ and $\mathbf{H}^{-1}_{B,S+1}$ using (29), (31) and (27), respectively, requiring nearly the same number of FPOs as that for removing a support vector. To update the errors over I in step 6, about $2|I|B$ FPOs are required, where $|I|$ denotes the size of set I. The total number of FPOs for processing the full set I in an iteration of procedure 2 from steps 1 to 6 is around $B[2N + |I|(|I| + 3B + 12)]$ with additional NB kernel evaluations.

4.3. Forward Incremental Basis Selection

Given a set of training data $\mathbf{X}$ and the number of basis vectors to be selected/optimized for the PSVM, denoted as n_{BS}. The PSVM with an optimal basis set is built using a two-stage procedure. In the first stage, an initial basis set of size n_{BS} is selected in a forward incremental procedure.

Note that as the size of the basis set increases, the precision of the PSVM model (over the training set) tends to improve and approach that of the full SVM, thus more training points are separated. If all the training points are separated during the incremental selection process, no more basis vectors are selected, even though $B < n_{BS}$. In other words, the forward incremental

selection process is terminated when a given number n_{BS} of basis vectors are selected, or the training set is fully separated.

A training point, say $\mathbf{x}_k$, is separated if $y_k f(\mathbf{x}_k) > 0$, or equivalently, $e_k < 1$, where $f(\mathbf{x}_k)$ is the discriminant function (16) evaluated at $\mathbf{x}_k$ for the current $\mathbf{w}_{B,S}$ and $b_{B,S}$, and e_k is the corresponding error of $\mathbf{x}_k$ as defined in (11).

Note again that the solution basis vector set needs to be identified by invoking procedure 2 for each change (an increase here) to the basis set, where the errors are always computed over the full training set. The training points that are separated can be easily counted based on these errors.

Procedure 3 - *Forward incremental selection for stage I*

Step 1 *Initialize*: invoke procedure 1 to initialize the following incremental forward selection procedure.

Step 2 *Loop Control*: if $B = n_{BS}$ or all the training points are separated, then terminate this procedure. Otherwise loop for the following two steps.

Step 3 *Increase*: evaluate the reduction in objective function value $\delta J_B^+(\bar{\mathbf{x}}_k)$ given in (44) for $\forall \bar{\mathbf{x}}_k \notin \overline{\mathbf{X}}_B$ that satisfies the independent condition (48). Look for the one, denoted as $\bar{\mathbf{x}}_{B+1}$, that gives the maximal reduction, and add it into the current basis set $\overline{\mathbf{X}}_B$. Correspondingly, compute $\mathbf{H}_{B+1,S}$, $\mathbf{w}_{B+1,S}$, $b_{B+1,S}$, $q_{B+1,S}$ and $\mathbf{l}_{B+1,S}$ using (40), (42) and (45), respectively. Finally let $B \leftarrow B + 1$.

Step 4 *Update*: with the increased basis set, invoke procedure 2 to identify the solution support vector set, where the errors for the training points are updated. Goto step 2 to select more basis vectors.

Scoring each candidate using (44) requires about $2B(B+4)+2n_{SV}(B+1)$ FPOs, totalling up to $2\kappa[B(B+4)+n_{SV}(B+1)]$ FPOs with additional $\kappa(n_{SV}+B)$ kernel evaluations, where $\kappa \leq N-B$ is the size of the candidate set. Each element from the candidate set is to be scored, and the one gaining the most scores is added to the basis set in each iteration of step 3. To add a selected basis vector into $\overline{\mathbf{X}}_B$ and compute $\mathbf{H}_{B+1,S}$, $\mathbf{w}_{B+1,S}$, $b_{B+1,S}$, $q_{B+1,S}$ and $\mathbf{l}_{B+1,S}$ using (40), (42) and (45) takes about $B(B+6)$ additional FPOs. The computational complexity of procedure 3 for incremental basis vector selection in the loop for $B = 1$ to n_{BV} is of order $O(N^2 + \kappa n_{SV} n_{BV}^2)$ FPOs

with additional kernel evaluations of order $O(N^2 + \kappa n_{SV} n_{BV})$, including one run of the initialization (procedure 1) and n_{BV} loops of procedure 3, given that $n_{BV} \ll n_{SV} \leq N$ and $|I| \ll n_{SV}$ generally holds, where n_{SV} and n_{BV} are the sizes of the support vector set and the final basis vector set, respectively.

Once the initial basis set is produced, it is then refined using a procedure that combines incremental selection and decremental pruning in the second stage.

4.4. Basis Set Refinement

To improve the computational efficiency, the basis vectors in $\overline{\mathbf{X}}_B$ are sorted by their significance, such that less significant basis vectors are checked (for possible replacement) first.

Procedure 4 *- Basis set refinement for stage II*

Step 1 *Initialize*: for the current basis set $\overline{\mathbf{X}}_B$, identify the candidate set $\mathbf{X}_{\widetilde{B}}$; sort the basis vectors such that $\delta J^-_{B,S}(\bar{\mathbf{x}}_1) \leq \cdots \leq \delta J^-_{B,S}(\bar{\mathbf{x}}_B)$. Let $i = 1$ as the counter.

Step 2 *Terminate*: if $i > B$ exit this procedure.

Step 3 *Check*: remove $\bar{\mathbf{x}}_i$ from $\overline{\mathbf{X}}_B$; produce the intermediate basis set $\overline{\mathbf{X}}_{B^{i)}}$. Compute $\delta J^+_{B^{i)},S}(\mathbf{x}_k), \forall \mathbf{x}_k \in \mathbf{X}_{\widetilde{B}}$. Find the most significant candidate, namely $\mathbf{x}_j = \arg\max\{\delta J^+_{B^{i)},S}(\mathbf{x}_k), \forall \mathbf{x}_k \in \mathbf{X}_{\widetilde{B}}\}$.

Step 4 *Shunt*: If the training set is fully separated, then goto step 5. If $\delta J^-_{B,S}(\bar{\mathbf{x}}_i) < \delta J^+_{B^{i)},S}(\mathbf{x}_j)$ then goto step 6; otherwise let $i \leftarrow i + 1$ and goto step 2.

Step 5 *Decrease*: let $\overline{\mathbf{X}}_{B-1} = \overline{\mathbf{X}}_{B^{i)}}$, $B \leftarrow B - 1$, and then goto step 7.

Step 6 *Increase*: add $\mathbf{x}_j$ into $\overline{\mathbf{X}}_{B^{i)}}$, let $\overline{\mathbf{X}}_B = [\overline{\mathbf{X}}^T_{B^{i)}}, \mathbf{x}^T_j]^T$ and then goto step 7.

Step 7 *Update*: invoke procedure 2 to identify the solver $\mathbf{X}_S$ for the updated basis set $\overline{\mathbf{X}}_B$. Goto step 1 to continue refining the basis set.

In procedure 4, all the basis vectors are checked one-by-one. If there is a basis vector, say $\bar{\mathbf{x}}_i \in \overline{\mathbf{X}}_B$, such that if replacing it with a vector outside $\overline{\mathbf{X}}_B$ cause a reduction in the objective function, then take the replacement. This checking process is repeated until no such basis vector exist. Effectively, in

the refined basis set obtained by procedure 4, any change in a basis vector will result in an increase (at least no decrease) in the objective function value. In this sense the refined basis set is locally optimized.

With regard to the computational complexity, evaluation of the significance of the B basis vectors $\delta J^{-}_{B,S}(\bar{\mathbf{x}}_i), i = 1, \cdots, B$ in step 1 is based on (58), requires $5B$ FPOs. In step 3, corresponding to the intermediate basis set $\overline{\mathbf{X}}_{B^{i)}}$, $\mathbf{H}_{B^{i)},S}$, $\mathbf{w}_{B^{i)},S}$, $b_{B^{i)},S}$, $\mathbf{l}_{B^{i)},S}$ and $b_{B^{i)},S}$, which are used for candidates significance evaluation using (44), must be computed (while the support vector set $\mathbf{X}_S$ keeps unchanged, as previously mentioned). This requires about $(B-1)(B+5)$ FPOs without any kernel evaluation.

Based on the previous complexity analysis for procedure 3, the overall computations for scoring a candidate set of size $\kappa, \kappa \leq N - B$ with the intermediate basis set $\overline{\mathbf{X}}_{B^{i)}}$ of $B-1$ basis vectors here is $2\kappa[(B-1)(B+3)+ n_{SV}B]$ FPOs with additional $\kappa(n_{SV}+B-1)$ kernel evaluations. Decreasing the basis set in step 5 involves no extra computation. Adding the most significant candidate into the intermediate basis set $\overline{\mathbf{X}}_{B^{i)}}$ of $B-1$ basis vectors here is the same as that in procedure 3, given by $(B-1)(B+5)$ FPOs here.

Finally identify the solution $\mathbf{X}_S$ for the updated basis set $\overline{\mathbf{X}}_B$ by invoking procedure 2, each run of which requires $B[2N+|I|(|I|+3B+12)]$ FPOs with additional NB kernel evaluations. The overall complexity of each iteration of the basis set refinement procedure is $O(Nn_{BV}+\kappa n_{SV}n_{BV}^2)$ with an additional $O(\kappa n_{SV} + Nn_{BV})$ kernel evaluations, given again that $n_{BV} \ll n_{SV} \leq N$ and $|I| \ll n_{SV}$ generally holds.

With regard to the memory requirement, the proposed algorithm needs to store the following quantities: The $n_{BV} \times n_{BV}$ matrix inverse $\mathbf{H}^{-1}_{B,S}$ defined in (15), the scalar $q_{B,S}$ and $n_{BV} \times 1$ vector $\mathbf{l}_{B,S}$ defined in (28), and the scalar $b_{B,S}$ and $n_{BV} \times 1$ vector $\mathbf{w}_{B,S}$ defined in (14). Note again that $\mathbf{H}^{-1}_{B,S}$ is symmetric, one needs only to store its triangular part. In addition, storing the N errors of the full training set is also required. Furthermore the indices of the n_{SV} support vectors $\mathbf{X}_S$, and the n_{BV} basis vectors $\overline{\mathbf{X}}_B$ need to be stored. The overall memory requirement of the PSVM algorithm is of order $O(n_{BV}^2 + N)$ additional to catching the kernel values.

5. Simulation Examples

In this section, the proposed algorithm, referred to as PSVM, is tested on seven public benchmark problems. The statistics/properties of the data

Table 1: Statistics of the benchmarking problems

Problem	Training set	Testing set	#Features /#Classes	$\bar{\sigma}^2$	Source
split	1000	2175	60 / 2	445	Delve [15]
adult	1605	30956	123 / 2	12.5	UCI [16]
webcat	2477	47272	300 / 2	57	LIBSVM [17]
colon	30	32	2000 / 2	2468.08	LIBSVM [17]
leukemia	38	34	7129 / 2	14997.7	LIBSVM [17]
dna	2000	1186	180 / 3	38	Statlog [18]
segment	1120	1190	19 / 7	8.5746	Statlog [18]

sets of these benchmarking problems are listed in Table 1. The experiments presented in this section were benchmarked against a PIII-800MHz CPU with 1GB RAM and Windows XP oerating system. There particular tests have been chosen because they have been publically available for some time, are ameanable to solution by standard libraries and have been used in our previous work to redily benchmark algorithm development [1] [13].

A well-known SVM package, LibSVM [17], and a direct sparse SVM algorithm (SpSVM) by Keerthi *et al.* (2006)[9] is employed for comparison. Note that the LibSVM package is coded in C/C++ and solves for standard SVMs by an SMO-style algorithm. SpSVM is almost the same as the RSVM methods by Lee and Mangasarian (2001)[5] and by Lin and Lin (2003)[6], with the only difference being the regularization term (for the maximal margin) in the objective functions. Both SpSVM and PSVM are coded in MATLAB and tested on each of the seven benchmark problems; the resulting sparse SVM models are compared with the corresponding standard SVM models produced by LibSVM.

Again it should also be noted that the algorithm we present here was developed with mobile computing applications in mind based on previous research [1] that used a less memory efficient version of this algorithm for activity classification. Thus for a more robust test of the new two stage algorithm we test on open data against well know SVM algorithms and libraries.

Note that SpSVM constructs a sparse SVM model in a forward incremental way by selecting one basis vector at a time from the training set outsize

Table 2: Results of LibSVM on the benchmark problems

Data set	C, γ (p, q)	#SVs	S.R. (%) training	S.R. (%) testing	Running time (s)
splice	2, 3.0	601	99.800	90.161	0.563
adult	3,−2.1	645	85.545	84.433	0.531
webcat	3, 1.2	311	99.112	98.083	0.594
colon	4,−2.7	20	100.000	75.000	0.235
leukemia	3,−0.6	31	100.000	82.353	1.110
dna	3,−0.6	1,553	99.800	96.206	2.641
segment	9, 2.7	386	99.554	97.311	0.281

the existing basis vector set. A Newton search based on Cholesky decomposition is performed to search for the optimal expansion coefficients and bias for each selection of an expansion vector. PSVM however, solves for the optimal normal vector and bias using a recursive method base on the closed solution (5), avoiding the inversion of large matrices. Furthermore, PSVM employs an additional stage to refine the basis set after the forward selection procedure. With this basis set refinement procedure, a locally optimal basis set is approached.

In this study, the following Gaussian kernel is employed for SVM classifiers.

$$K(\mathbf{x}, \bar{\mathbf{x}}) = \exp(-\gamma \|\mathbf{x} - \bar{\mathbf{x}}\|_2^2), \gamma > 0 \tag{71}$$

For each of LibSVM, SpSVM and PSVM, the best combination of the penalty coefficient C and the kernel parameter γ is identified by a grid-search for $C \in \{2^p, p = -4, -3, \cdots, 15\}$ and $\gamma \in \{\frac{2^q}{2\bar{\sigma}^2}, q = -4.5, -4.2, \cdots, 4.5\}$, resulting in 620 combinations, where $\bar{\sigma}^2 = \frac{1}{2}(\sigma_{min}^2 + \sigma_{max}^2)$, σ_{min} and σ_{max} are respectively the minimal and maximal l^2-norms of all the points (including training and testing sets). For each of the problems, $\bar{\sigma}^2$ is listed in table 1. For multi-class problems we consider that C and γ settings for all the involved binary SVMs via one-versus-the-rest scheme [19] are the same.

The results of LIBSVM are listed in table 2 with the success rates on both the training set (S.R. training) and the testing set (S.R. testing), and the running time (in seconds) for each of the benchmark problems listed in table 1. The identified optimal combinations of parameters C and γ are

Table 3: Results of PSVM on the benchmark problems

Data set	C, γ (p, q)	#BVs	S.R. (%) training	S.R. (%) testing	Running time (s)
splice	3, 3.3	48	90.600	89.011	6.875
adult	3, −0.3	28	85.109	84.559	7.671
webcat	11, 0.6	14	98.345	97.861	7.390
colon	12, 1.5	3	96.667	78.125	0.250
leukemia	11, 0.6	4	100.000	91.176	1.094
dna	7, −1.5	144	97.600	95.363	33.890
segment	13, 1.8	77	98.304	97.479	6.610

Table 4: Results of SpSVM on the benchmark problems

Data set	C, γ (p, q)	#BVs	S.R. (%) training	S.R. (%) testing	Running time (s)
splice	4, 3.9	50	89.900	88.460	9.672
adult	1, 0.6	22	84.735	84.407	12.516
webcat	13, 3.0	9	97.860	97.737	19.656
colon	10, −0.6	3	96.667	78.125	0.047
leukemia	12, −0.6	3	97.368	91.176	0.156
dna	12, 0.9	138	97.300	94.941	82.281
segment	12, 0.3	182	96.696	95.294	9.609

given by (p, q) as previously mentioned. The success rate is defined as the percentage of points of a data set for which a model successfully predicted the labels. The number of support vectors (#SVs) indicates the complexity of the resulting regular SVM classification models.

The results of using PSVM and SpSVM are listed in tables 3 and 4, respectively. The complexity of the resulting sparse SVM classification models are indicated by #BVs, the number of the basis vectors, which is determined by gradually increasing the model size until no significant improvement in the performance when applied on the testing set in obtained, measured by

the testing success rate (S.R. testing).

Note that the last two problems (dna and segment) are 3- and 7-classes problems, respectively. The one-versus-the-rest scheme [19] employed here uses multiple (3 and 7 respectively) binary SVMs to separate the more than two classes. The model size is the sum of sizes of the involved SVMs; the running time is the total time taken to produce all the involved SVMs. Note again that, if all the training points are separated, PSVM will not increase the model size in the first stage, while a further decrease is possible in the second stage (see procedures 3 and 4 for the first and the second stages listed previously). The resulting models produced by PSVM can be smaller than specified, particularly for multi-class problems.

Compared with LibSVM, the proposed PSVM algorithm can produce SVM models of much smaller sizes while of competitive performance on test data. The sizes of PSVM models can be 5 to 20 percent of the size of standard SVM models.

Compared with the SpSVM, the proposed PSVM implements a basis set refinement procedure in addition to a forward incremental procedure, and the basis vector set is locally optimized. Test results in tables 3 and 4 show that PSVM produces models of similar sizes to that of SpSVM. However PSVM-models are of slightly better or equivalent performance to SpSVM-models in all the cases with regard to success rates in both training and testing data sets.

With regard to algorithm complexity, the proposed PSVM algorithm recursively checks (for linear independency) and evaluates candidate basis vectors, and update the expansion coefficients and the bias for changes in the basis vector set. Inversion of large matrices and continuous search for the expansion coefficients and the bias are avoided. The running times taken by PSVM are significantly shorter that SpSVM to produce models of similar sizes for relatively large sets, benefiting from the efficient basis vector scoring method/algorithm of closed form. However for problems of small training sets (colon and leukemia), SpSVM is more efficient than PSVM, given that SpSVM selects the basis set using an incremental procedure, while PSVM refines the basis set using a procedure additional to the incremental selection procedure similar to that of SpSVM.

6. Conclusion

In this paper, an algorithm for a projected support vector machine (PSVM) is proposed. A basis vector set is selected in the kernel-induced feature space. The training points are projected onto the subspace spanned by the selected basis vector set. With the projected training points on the selected basis set, a standard SVM is then produced in the subspace, referred to as the PSVM. As the size of the expansion of the PSVM discriminant function equals the size of the selected basis set, the testing complexity of the produced PSVM model can be controlled.

Furthermore, since solving for the support vector set and the corresponding elements of the PSVM, and optimizing the basis set are two closely coupled problems. Any change in the basis set may result in a different PSVM solution, This helps ensure that even when only a subspace is selected that then resulting SVM also provides a good performance. These two problems are solved alternatively to produce PSVMs with an optimal basis set $\overline{\mathbf{X}}_B$ (of a given size B) selected from the training point set.

A recursive algorithm is derived for sequential refinements of the basis set using an efficient basis vector scoring method of closed form. Based on this a two-stage algorithm is described which achieves a locally optimal basis set. In the first stage a forward incremental procedure is employed to select basis vectors from the training set, resulting in an initial model of a basis set of specified size. This initial model is refined in the second stage, where each basis vector is compared with those training points (vectors) not included in the current basis set. If there exists a training point which is superior to the basis vector, then the basis vector is replaced by the superior training point. This refinement procedure is iterated until no outstanding point in the training set is superior over any one in the selected basis set, hence a locally optimal basis set.

To guarantee the linear independence, a condition that a vector can be selected as a new basis vector is proposed. This condition is easy to check/implement in the recursive context.

The proposed PSVM has been tested on seven public benchmark classification problems and compared with an existing sparse primal SVM algorithm (SpSVM) and LibSVM that produce standard SVMs. Test results shown that models produced by the proposed PSVM are of equivalent performance to standard SVMs, whilst being much smaller in model sizes (only 5 to 20 percent of kernel evaluations in the model expansion).

Compared with the existing SpSVM, the proposed PSVM solves for SVMs in the subspace recursively, with neither inversion of large matrices nor continuous searches for the expansion coefficients and bias as is done in SpSVM. The proposed PSVM is significantly faster than SpSVM for large data sets, benefiting from the efficient method for candidate basis vectors scoring.

The properties of the proposed PSVM algorithm make it much more suitable for implementation in devices which have limited memory and processing resources (e.g. mobile and embedded software applications). This makes the algorithm suitable for activity recognition using mobile platforms where there are a large range of applications in assisted living.

Acknowledgement

This work was part-funded by the European Commission under the Seventh Framework Programme: large-scale integrating project *HaptiMap*, FP7-ICT-224675.

References

[1] J.-X. Peng, S. Ferguson, K. Rafferty and P. D. Kelly, An efficient feature selection method for mobile devices with application to activity recognition, Neurocomputing, 74 (2011), 3543-3552

[2] V. Vapnik and A. Lerner, Pattern recognition using generalized portrait method,Automation and Remote Control 24(1963),774-780.

[3] V.N. Vapnik, Estimation of Dependences Based on Empirical Data, Addendum 1, New York: Springer-Verlag, 1982.

[4] T. Downs, K. E. Gates, and A. Masters, Exact simplification of support vector solutions, Journal of Machine Learning Research, 2(2001), 293-297.

[5] Y. J. Lee and O. L. Mangasarian, RSVM: Reduced support vector machines, Proceedings of the SIAM International Conference on Data Mining ,http://dx.doi.org/10.1137/1.9781611972719.13, (2001), 1-16.

[6] K. Lin and C. Lin, A study on reduced support vector machines, IEEE Transactions on Neural Networks, 14(2003), 1449-1459.

[7] M. Wu, B. Schölkopf, and G. Bakir, Building sparse large margin classifiers, Proc. 22nd Int. Conf. Mach. Learn., (2005), 996-1003.

[8] M. Wu, B. Scholkopf, and B. Bakir, A direct method for building sparse kernel learning algorithms, J. Mach. Learn. Res., 7(2006), 603-624.

[9] S. S. Keerthi, O. Chapelle and D. DeCoste, Building support vector machines with reduced classifier complexity, Journal of Machine Learning Research, 7(2006), 1493-1515.

[10] D. Anguita, A. Ghio, S. Pischiutta and S. Ridella, A hardware friendly support vector machine for embedded automotive applications, International Joint Conference on Neural Networks,(2007), 121-167.

[11] D. Anguita, A. Ghio, Oneto L., X. Parra, J. Reyes-Ortiz, Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine, Lecture Notes in Computer Science Ambient Assisted Living and Home Care, (2012), 216-223.

[12] I. Steinwart, Sparseness of support vector machines - some asymptotically sharp bounds, Proceedings of the 16th NIPS Conference, (2004), 169-184.

[13] J.-X. Peng, S. Ferguson, K. Rafferty, V. Stewart, A sequential algorithm for sparse support vector classifiers, Pattern Recognition, (2013), 46(4), 1195-1208.

[14] B. Noble and J. W. Daniel. Applied Linear Algebra. 3rd Edition, Prentice-Hall, 1988.

[15] Delve Datasets, The University of Toronto, Toronto, Ontario, Canada. [Online] http://www.cs.toronto.edu/∼delve/data/datasets.html

[16] C. L. Blake and C. J. Merz, UCI Repository of Machine Learning Databases. (1998) Univ. California, Dept. Inform. Comput. Sci., Irvine, CA. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[17] C.-C. Chang and C.-J. Lin., LIBSVM: a library for support vector machines, (2001), Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

[18] D. Michie, D. J. Spiegelhalter, and C. C. Taylor (1994), Machine Learning, Neural and Statistical Classification, Online available: http://www1.maths.leeds.ac.uk/∼charles/statlog/whole.pdf

[19] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, Large margin DAGs for multiclass classification, Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press, 12 (2000), 547-553.